计算机基础与实训教材系列

ASP.NET 4.0动态网站开发实用教程

杨春元　编著



（理论→实例→上机→习题）4阶段教学模式

任务驱动的讲解方式，方便学习和教学

众多典型的实例操作，注重培养动手能力

PPT电子教案及素材免费下载，专业的网上技术支持

清华大学出版社

计算机基础与实训教材系列

ASP.NET 4.0 动态网站开发实用教程

杨春元　编著

清华大学出版社
北　京

内 容 简 介

本书由浅入深、循序渐进地介绍了使用 ASP.NET 4.0 开发动态网站的基本知识和使用技巧。全书共分 11 章，分别介绍了 ASP.NET 的发展历程，VWD 2010 集成开发环境，ASP.NET 的内置对象和配置文件，各种服务器控件的使用，CSS 样式、主题和母版页的使用，使用 ADO.NET 访问和操纵数据库，LINQ 查询技巧，ASP.NET AJAX，XML 和 Web 服务，使用 jQuery 美化网页以及网站的部署与发布。最后一章还安排了项目实践，综合运用所学知识创建一个简易的 BBS 网站，提高和拓宽读者的实际技能。

本书内容丰富，结构清晰，语言简练，图文并茂，具有很强的实用性和可操作性，是一本适合于大中专院校、职业院校及各类社会培训学校的优秀教材，也是广大初、中级电脑用户的自学参考书。

本书对应的电子教案、实例源文件和习题答案可以到 http://www.tupwk.com.cn/edu 网站下载。

图书在版编目(CIP)数据

ASP.NET 4.0 动态网站开发实用教程/杨春元 编著. —北京：清华大学出版社，2012.3
(计算机基础与实训教材系列)

ISBN 978-7-302-27799-6

Ⅰ. ①A… Ⅱ. ①杨… Ⅲ. ①网页制作工具—程序设计—教材 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2011)第 282441 号

责任编辑：胡辰浩　袁建华
装帧设计：牛艳敏
责任校对：成凤进
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn, 010-62796045
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　张**：21.75　　**字　数**：571 千字
版　　次：2012 年 3 月第 1 版　　**印　次**：2012 年 3 月第 1 次印刷
印　　数：1～5000
定　　价：36.00 元

产品编号：

编审委员会

计算机基础与实训教材系列

丛 书 序

计算机已经广泛应用于现代社会的各个领域，熟练使用计算机已经成为人们必备的技能之一。因此，如何快速地掌握计算机知识和使用技术，并应用于现实生活和实际工作中，已成为新世纪人才迫切需要解决的问题。

为适应这种需求，各类高等院校、高职高专、中职中专、培训学校都开设了计算机专业的课程，同时也将非计算机专业学生的计算机知识和技能教育纳入教学计划，并陆续出台了相应的教学大纲。基于以上因素，清华大学出版社组织一线教学精英编写了这套“计算机基础与实训教材系列”丛书，以满足大中专院校、职业院校及各类社会培训学校的教学需要。

一、丛书书目

本套教材涵盖了计算机各个应用领域，包括计算机硬件知识、操作系统、数据库、编程语言、文字录入和排版、办公软件、计算机网络、图形图像、三维动画、网页制作以及多媒体制作等。众多的图书品种可以满足各类院校相关课程设置的需要。

⊙ 已出版的图书书目

《计算机基础实用教程》	《中文版 Excel 2003 电子表格实用教程》
《计算机组装与维护实用教程》	《中文版 Access 2003 数据库应用实用教程》
《五笔打字与文档处理实用教程》	《中文版 Project 2003 实用教程》
《电脑办公自动化实用教程》	《中文版 Office 2003 实用教程》
《中文版 Photoshop CS3 图像处理实用教程》	《JSP 动态网站开发实用教程》
《Authorware 7 多媒体制作实用教程》	《Mastercam X3 实用教程》
《中文版 AutoCAD 2009 实用教程》	《Director 11 多媒体开发实用教程》
《AutoCAD 机械制图实用教程(2009 版)》	《中文版 Indesign CS3 实用教程》
《中文版 Flash CS3 动画制作实用教程》	《中文版 CorelDRAW X3 平面设计实用教程》
《中文版 Dreamweaver CS3 网页制作实用教程》	《中文版 Windows Vista 实用教程》
《中文版 3ds Max 9 三维动画创作实用教程》	《电脑入门实用教程》
《中文版 SQL Server 2005 数据库应用实用教程》	《中文版 3ds Max 2009 三维动画创作实用教程》
《中文版 Word 2003 文档处理实用教程》	《Excel 财务会计实战应用》
《中文版 PowerPoint 2003 幻灯片制作实用教程》	《中文版 AutoCAD 2010 实用教程》
《中文版 Premiere Pro CS3 多媒体制作实用教程》	《AutoCAD 机械制图实用教程(2010 版)》
《Visual C#程序设计实用教程》	《Java 程序设计实用教程》

(续表)

《Mastercam X4 实用教程》	《SQL Server 2008 数据库应用实用教程》
《网络组建与管理实用教程》	《中文版 3ds Max 2010 三维动画创作实用教程》
《中文版 Flash CS3 动画制作实训教程》	

⊙ 即将出版的图书书目

《Oracle Database 11g 实用教程》	《中文版 Pro/ENGINEER Wildfire 5.0 实用教程》
《ASP.NET 3.5 动态网站开发实用教程》	《中文版 Office 2007 实用教程》
《AutoCAD 建筑制图实用教程（2009 版）》	《中文版 Word 2007 文档处理实用教程》
《中文版 Photoshop CS4 图像处理实用教程》	《中文版 Excel 2007 电子表格实用教程》
《中文版 Illustrator CS4 平面设计实用教程》	《中文版 PowerPoint 2007 幻灯片制作实用教程》
《中文版 Flash CS4 动画制作实用教程》	《中文版 Access 2007 数据库应用实例教程》
《中文版 Dreamweaver CS4 网页制作实用教程》	《中文版 Project 2007 实用教程》
《中文版 Indesign CS4 实用教程》	《中文版 CorelDRAW X4 平面设计实用教程》
《中文版 Premiere Pro CS4 多媒体制作实用教程》	《中文版 After Effects CS4 视频特效实用教程》

二、丛书特色

1、选题新颖，策划周全——为计算机教学量身打造

本套丛书注重理论知识与实践操作的紧密结合，同时突出上机操作环节。丛书作者均为各大院校的教学专家和业界精英，他们熟悉教学内容的编排，深谙学生的需求和接受能力，并将这种教学理念充分融入本套教材的编写中。

本套丛书全面贯彻“理论→实例→上机→习题”4 阶段教学模式，在内容选择、结构安排上更加符合读者的认知习惯，从而达到老师易教、学生易学的目的。

2、教学结构科学合理，循序渐进——完全掌握“教学”与“自学”两种模式

本套丛书完全以大中专院校、职业院校及各类社会培训学校的教学需要为出发点，紧密结合学科的教学特点，由浅入深地安排章节内容，循序渐进地完成各种复杂知识的讲解，使学生能够一学就会、即学即用。

对教师而言，本套丛书根据实际教学情况安排好课时，提前组织好课前备课内容，使课堂教学过程更加条理化，同时方便学生学习，让学生在学习完后有例可学、有题可练；对自学者而言，可以按照本书的章节安排逐步学习。

3、内容丰富、学习目标明确——全面提升“知识”与“能力”

本套丛书内容丰富，信息量大，章节结构完全按照教学大纲的要求来安排，并细化了每一章内容，符合教学需要和计算机用户的学习习惯。在每章的开始，列出了学习目标和本章重点，便于教师和学生提纲挈领地掌握本章知识点，每章的最后还附带有上机练习和习题两部分内容，教师可以参照上机练习，实时指导学生进行上机操作，使学生及时巩固所学的知识。自学者也可以按照上机练习内容进行自我训练，快速掌握相关知识。

4、实例精彩实用，讲解细致透彻——全方位解决实际遇到的问题

本套丛书精心安排了大量实例讲解，每个实例解决一个问题或是介绍一项技巧，以便读者在最短的时间内掌握计算机应用的操作方法，从而能够顺利解决实践工作中的问题。

范例讲解语言通俗易懂，通过添加大量的“提示”和“知识点”的方式突出重要知识点，以便加深读者对关键技术和理论知识的印象，使读者轻松领悟每一个范例的精髓所在，提高读者的思考能力和分析能力，同时也加强了读者的综合应用能力。

5、版式简洁大方，排版紧凑，标注清晰明确——打造一个轻松阅读的环境

本套丛书的版式简洁、大方，合理安排图与文字的占用空间，对于标题、正文、提示和知识点等都设计了醒目的字体符号，读者阅读起来会感到轻松愉快。

三、读者定位

本丛书为所有从事计算机教学的老师和自学人员而编写，是一套适合于大中专院校、职业院校及各类社会培训学校的优秀教材，也可作为计算机初、中级用户和计算机爱好者学习计算机知识的自学参考书。

四、周到体贴的售后服务

为了方便教学，本套丛书提供精心制作的 PowerPoint 教学课件(即电子教案)、素材、源文件、习题答案等相关内容，可在网站上免费下载，也可发送电子邮件至 wkservice@vip.163.com 索取。

此外，如果读者在使用本系列图书的过程中遇到疑惑或困难，可以在丛书支持网站(http://www.tupwk.com.cn/edu)的互动论坛上留言，本丛书的作者或技术编辑会及时提供相应的技术支持。咨询电话：010-62796045。

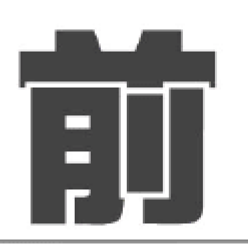
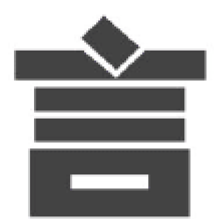
推荐课时安排

章 名	重 点 掌 握 内 容	教 学 课 时
第 1 章 ASP.NET 4.0 入门	1. 网站建设基础知识 2. VWD 2010 开发环境 3. 新建 Web 站点 4. ASP.NET 应用程序的工作原理	2 学时
第 2 章 ASP.NET 基础知识	1. ASP.NET 应用程序目录结构 2. ASP.NET 页面往返机制和生命周期 3. 创建和使用常用内置对象的方法 4. ASP.NET 的配置管理	3 学时
第 3 章 ASP.NET 服务器控件	1. 服务器控件的工作原理 2. 服务器控件的基类和常用事件 3. 列表控件的使用 4. 验证控件的用法 5. 导航控件 6. 登录控件 7. 创建并使用用户控件 8. 如何关闭视图状态	4 学时
第 4 章 样式、主题和母版页	1. 什么是 CSS 样式 2. VWD 中创建和使用 CSS 的工具 3. 主题与外观 4. skinID 属性 5. 母版页与内容页	3 学时
第 5 章 显示和操作数据库	1. 使用 SQL 操纵数据库表 2. ADO.NET 访问数据库 3. 数据绑定 4. 数据源控件 5. 数据控件 6. 事务处理技术	3 学时
第 6 章 使用 LINQ	1. LINQ 的各种形式及其适用场合 2. ADO.NET Entity Framework 3. 使用 EntityDataSource 控件来访问 EF 4. ListView 控件和 DataPage 控件的使用	2 学时



(续表)

章　名	重 点 掌 握 内 容	教 学 课 时
第 7 章　ASP.NET AJAX	1. ASP.NET AJAX 的基本知识 2. 使用 UpdatePanel 控件 3. 使用 UpdateProgress 控件 4. 使用 Timer 控件 5. 使用客户端 ASP.NET AJAX Library	3 学时
第 8 章　XML 和 Web 服务	1. XML 的基本结构 2. 使用 ADO.NET 访问 XML 3. 创建 Web 服务 4. 调用 Web 服务 5. 支持 AJAX 的 Web 服务	3 学时
第 9 章　jQuery 入门	1. jQuery 选择器 2. jQuery 筛选器 3. 使用 jQuery 增强页面 4. 使用 jQuery 插件 5. 编写 jQuery 插件 6. jQuery 与 Ajax	3 学时
第 10 章　部署 Web 站点	1. 发布前的准备工作 2. 复制 Web 站点 3. 在 IIS 下运行站点 4. 将数据移到远程服务器	1 学时
第 11 章　项目与实践	1. 需求分析 2. 数据库设计 3. 数据实体类设计 4. 母版页设计 5. 功能页面设计与实现	2 学时

ASP.NET 是 Microsoft 公司推出的基于.NET Framework 的 Web 应用开发平台，是 Web 应用开发的主流技术之一。使用 ASP.NET 进行 Web 应用开发，程序结构更加清晰，开发流程更加简单，从而可以提高开发效率，缩短开发周期。ASP.NET 4.0 是在 ASP.NET 3.5 的基础之上构建的，保留了其中很多令人喜爱的功能，并增加了一些其他领域的新功能和工具。VWD 是专门为构建 ASP.NET Web 站点而开发的，其中包含了大量有助于快速创建复杂 ASP.NET Web 应用程序的工具。

本书从教学实际需求出发，合理安排知识结构，从零开始、由浅入深、循序渐进地讲解了 ASP.NET 4.0 的基本知识和使用方法。本书共分为 11 章，主要内容如下：

第 1 章介绍了网站建设的基础知识、ASP.NET 的发展过程、VWD 2010 开发环境，以及 ASP.NET 的工作原理。

第 2 章介绍了 ASP.NET 的一些基础知识，学习和掌握这些知识是以后进行 ASP.NET 程序开发的基础和前提。主要包括 ASP.NET 的页面框架和页面类，ASP.NET 的内置对象以及 ASP.NET 的配置文件 Web.config 和全局文件 Global.asax。

第 3 章介绍了 ASP.NET 服务器控件的基本用法以及不同类别控件的功能。使用 ASP.NET 服务器控件，可以大幅减少开发 Web 应用程序所需编写的代码量，提高开发效率和 Web 应用程序的性能。

第 4 章介绍了 CSS 样式、主题和母版页。这些技术对于创建具有一致外观的网站非常有用，也有利于使站点看起来更专业和有吸引力。

第 5 章介绍了数据库的基本知识和 SQL 语言、ADO.NET 访问数据库的方法以及 ASP.NET 提供的数据绑定技术和数据控件的使用。

第 6 章介绍了 LINQ 语言及其语法，以及在 ASP.NET 项目中使用 LINQ 数据的许多方法。

第 7 章介绍了 ASP.NET AJAX 的基本知识，详细讲解了 ASP.NET AJAX 服务器控件的使用方法。

第 8 章介绍了 XML 和 Web 服务的基本概念以及如何创建和调用 Web 服务，包括在 AJAX 站点中使用 Web 服务。

第 9 章介绍了 jQuery 的基本语法和具体应用。jQuery 是继 Prototype 之后又一个优秀的 JavaScript 框架，jQuery 能够改变开发人员编写 JavaScript 脚本的方式，降低学习和使用 Web 前端开发的复杂度，提高网页开发效率，这无论是对于 JavaScript 初学者，还是对于 Web 开发资深专家，jQuery 都是必备的工具。

第 10 章介绍了 Web 应用程序的部署，包括复制 Web 站点、在 IIS 下运行站点和将数据库移动到远程服务器。

第 11 章介绍了一个具体的项目——BBS，综合运用了全书所学内容，实际开发一个小型的论坛网站。

本书图文并茂，条理清晰，通俗易懂，内容丰富，在讲解每个知识点时都配有相应的实例，方便读者上机实践。同时在难于理.解和掌握的部分内容上给出相关提示，让读者能够快速地提高操作技能。此外，本书配有大量综合实例和练习，让读者在不断的实际操作中更加牢固地掌握书中讲解的内容。

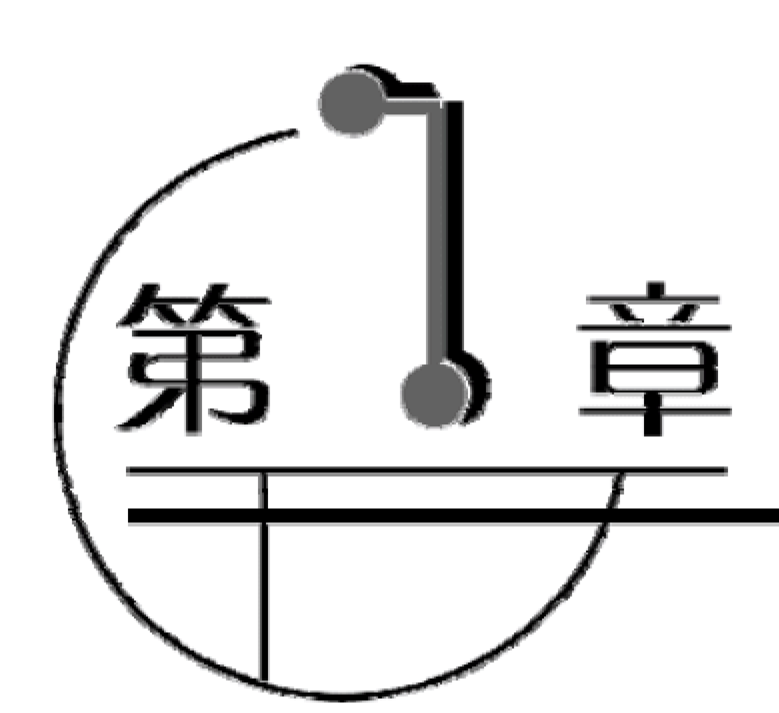
本书是集体智慧的结晶，参加本书编写的人员还有周高翔、宋友杰、昝舒馨、任运成、高晓红、张旭、孙成洪、余枭灵、张晓菊、卢华林、褚德华等。编写本书的过程中参考了相关文献，在此对这些文献的作者深表感谢。由于作者水平有限，本书不足之处在所难免，欢迎广大读者批评指正。我们的邮箱是：huchenhao@263.net，电话：010-62796045。

作者

2011 年 12 月

目录





第1章 ASP.NET 4.0 入门

学习目标

ASP.NET 4.0 是微软公司为了迎接网络时代的到来而提出的一个 Web 开发模型，它是建立在公共语言运行库上的编程框架。本章主要介绍了网站建设的基础知识、ASP.NET 的发展过程、VWD 2010 开发环境，以及 ASP.NET 的工作原理。

本章重点

- 网站建设基础知识
- VWD 2010 开发环境
- 新建 Web 站点
- ASP.NET 应用程序的工作原理

1.1 网站建设基础知识

互联网于 20 世纪 60 年代末出现，早期的互联网用户大多限于教育和国防机构。随着越来越多的用户在全球范围内实现信息共享，互联网逐渐兴盛起来。互联网的快速发展给人们的工作、学习和生活带来了重大变化，极大地提高了工作效率。在互联网的发展过程中，Web 网站的开发技术也得到了不断发展，最为关键的技术之一就是网站建设技术。本节将向读者介绍网站制作过程中静态网站、动态网站等的一些基本概念。

1.1.1 静态网站

早期的网站一般都是采用静态网页技术制作的静态网站。在静态网站中所有的内容都用 HTML 语言编写，存储在静态网页文件中，文件扩展名为.htm、.html、.shtml、.xml 等。虽然网

页中可以包含 GIF 动画、Flash 动画、滚动字幕等动态视觉效果，但这些动态效果只是视觉上的，都需要在服务器端以手工方式进行变换。这里所讨论的静态网站中的“静”是指网页内容在用户发出请求之前就已经生成了(也就是说，用户每次总能看到相同的页面)，Web 服务器只负责保存和传递 HTML 文件，而不进行额外处理，用户只能阅读网站所提供的信息。这种页面的请求模式如图 1-1 所示。

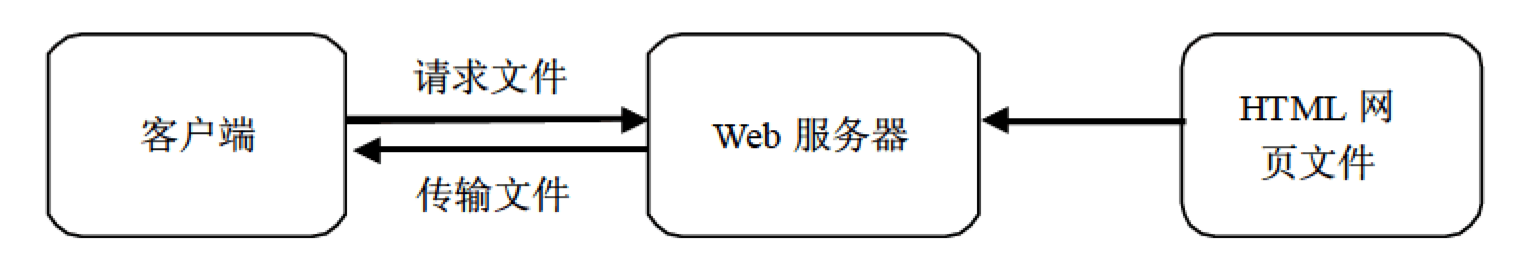


图 1-1　静态网页的请求模式

静态网站中网页的内容相对稳定，不需要通过数据库工作，对于 Web 服务器来说，处理负担不大，因此静态网站具有容易被搜索引擎检索、访问速度比较快等优点。

静态网站的致命弱点是不易维护，为了不断更新网页内容，网站管理者必须不断地重复制作 HTML 文件，随着网站内容和信息量的日益增长，维护工作将变得十分复杂。因此，静态网站往往适用于数据不多，网页比较固定，更新不频繁的情况。



1.1.2　动态网站

什么是动态网站呢？所谓“动”，并不是指网页上的 GIF 等动画图片，而是指用户与网站的交互性和互动性。动态网站一般应满足以下特征。

1. 交互性

动态网站中的网页会根据用户的要求和选择而改变和响应。网站管理员只需要掌握计算机基本操作方法，就可以方便、及时地更新网站内容；浏览网站的用户可以在网站中进行查询、留言等操作。可见，动态网站技术大大增加了用户与网站的交互性。

2. 在服务器端运行，方便更新

在服务器端运行的程序、网页、组件，会随不同用户、不同要求返回不同的页面，网站管理员无须手动更新网页文档，可以大大节省网站管理的工作量，如图 1-2 所示。

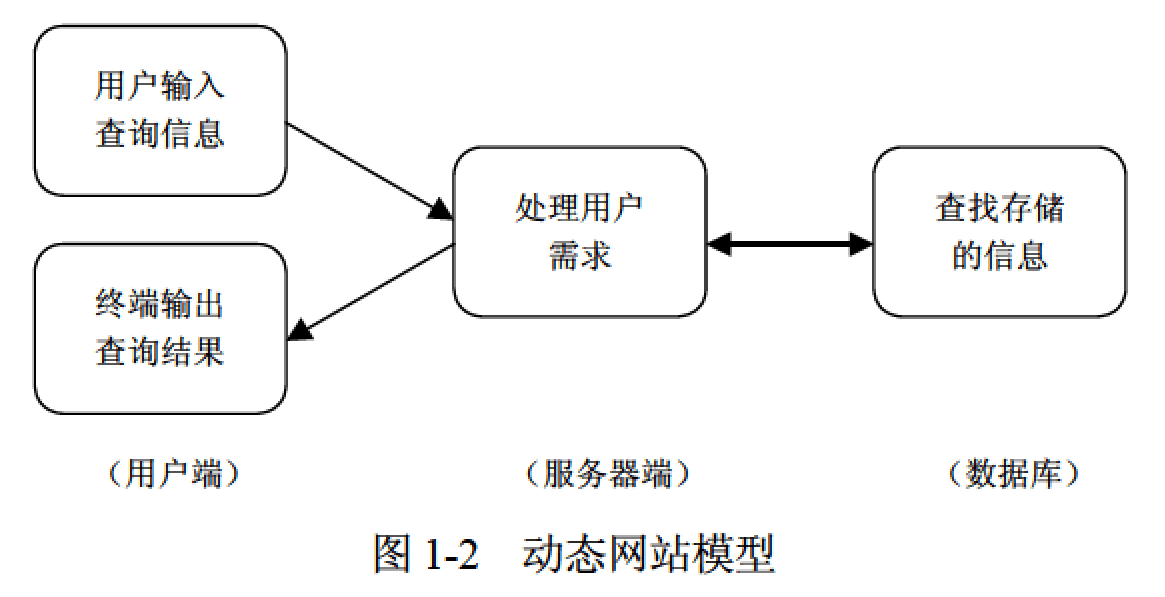


图 1-2　动态网站模型



3. 通过数据库进行架构

在动态网站中，网络管理员除了要设计网页视觉效果外，还要设计数据库和程序代码来使网站具有更多自动的和高级的功能。例如，购物网站中含有大量的商品种类和数量信息，为了方便查找，则应搭建数据库平台，以在网页上实现自动搜索。现在广泛使用的网上交易系统、在线采购系统、商务交流系统等都是由数据库提供技术支持的。

由上述特征可以看出，静态网站和动态网站的主要区别在于：静态网站的内容是在用户发出请求之前就预先生成的，而动态网站的内容则是在用户发出请求之后才产生。

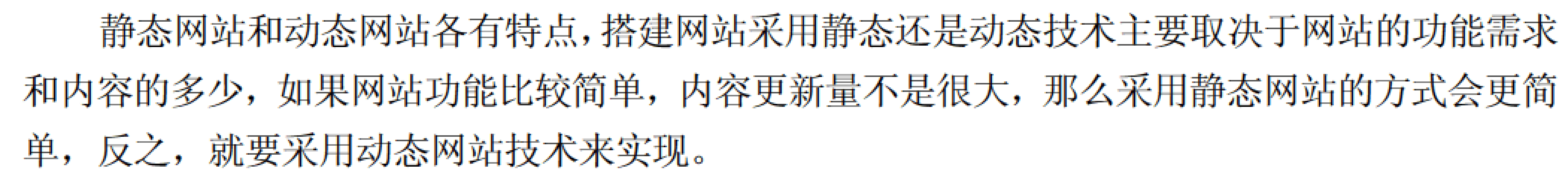
动态网站在发出请求之后生成内容有以下两个明显的优点：

- 首先，服务器端可以根据用户提交的请求以及请求中的信息来生成页面的内容。例如，在一个电子商务网站提交用户名和密码，那么用户将看到的下一个页面就是动态生成的页面，它包含了用户的私有帐户信息。
- 其次，服务器端可以根据最新的可用信息设置它所生成的页面内容。例如，很多网站都有显示当前在线用户数。在线用户数是实时信息，是在 Web 服务器接收用户请求时获取的。



静态网站和动态网站各有特点，搭建网站采用静态还是动态技术主要取决于网站的功能需求和内容的多少，如果网站功能比较简单，内容更新量不是很大，那么采用静态网站的方式会更简单，反之，就要采用动态网站技术来实现。

静态网站可以使用 Frontpage 或 Dreamweaver 等网页编辑工具来建立，而动态网站则需要使用服务器端网页技术，如本书介绍的 ASP.NET 来搭建。



1.2 ASP.NET 与 VWD 2010

自从在 2002 年初首次发布.NET Framework 1.0 以来，Microsoft 花了大量精力和时间来开发 ASP.NET，它是.NET Framework 的一部分，可以用来构建 Web 应用程序。本节将介绍 ASP.NET 的发展历史及其开发环境。



1.2.1 ASP.NET 的历史

早期的 Web 程序开发是十分繁琐的事情，一个简单的动态页面就需要编写大量的代码(一般用 C 语言)才能完成。

1996 年，Microsoft 推出了 ASP(Active Server Page，活动服务器页面，现在人们常称之为传统 ASP)1.0 版。它允许采用 VBScript/JavaScript 这些简单的脚本语言编写代码，允许将代码直接嵌入 HTML 中，从而使得设计动态 Web 页面的工作变得简单。ASP 能够通过内置的组件实现强大的功能(如 Cookie)。ASP 最显著的贡献就是推出了 ActiveX Data Objects(ADO)，它使得程序对数据库的操作变得十分简单。

1998 年，微软发布了 ASP 2.0 和 IIS 4.0。与前一版相比，2.0 版最大的改进是外部的组件需要初始化。用户能够利用 ASP 2.0 和 IIS 4.0 构建各种 ASP 应用，而且每个组件有了自己单独的内存空间，可以进行事务处理。

随后，微软在 Windows 2000 Server 系统中提供了 IIS 5.0 和 ASP 3.0。此次升级，最主要的改变就是把很多事情交给 COM+来做，效率比以前的版本更高，而且更稳定。

ASP.NET 1.0 的发布意味着从过去的 Microsoft 技术向构建 ASP Web 站点的飞跃。ASP.NET 1.0 在结构上与传统的 ASP 版本截然不同，几乎完全是基于组件和模块化的。ASP.NET 1.0 及相关的 Visual Studio .NET 2002 的引入给开发人员带来了如下好处：

- 页面显示与代码清楚地分开。使用传统 ASP 时，编程逻辑常常散布在整个页面的 HTML 中，使得后面对页面的修改比较困难。
- 开发模型更接近于桌面应用程序的编程方式。这样很多 Visual Basic 程序员可以轻松地转换到 Web 应用程序开发。
- 有一个功能丰富的开发工具(称为 Visual Studio .NET)，开发人员可以通过它可视化地创建和编写 Web 应用程序代码。
- 有几种面向对象的编程语言可供选择，其中 Visual Basic .NET 和 C#(读作 C-Sharp)是目前最流行的两种语言。
- 它可以访问整个.NET Framework，这意味着 Web 开发人员首次拥有了一种统一且容易的方式，来使用数据库、文件、网络工具等许多高级功能。



知识点

ASP.NET 提出了代码隐藏类(CodeBehind)的概念，把逻辑代码(.aspx.cs)和表现页面(.aspx)分离开来，使用户很容易使用后台代码来控制页面的逻辑功能。

2003 年，Microsoft 公司发布了 Visual Studio .NET 2003，提供了在 Windows 操作系统下开发各类基于.NET 框架的全新的应用程序开发平台(称为.NET 1.1)。

2005 年 11 月，Microsoft 发布了 Visual Studio 2005 和 ASP.NET 2.0。它修正了以前版本中的一些 Bug 并在移动应用程序开发、代码安全以及对 Oracle 数据库和 ODBC 的支持等方面都做了很多改进。

Microsoft 公司在 2007 年 11 月发布的 Visual Studio 2008 和 ASP.NET 3.5 中添加了一系列很酷的新功能，主要的新功能包括 LINQ 以及 AJAX 框架整合。2008 年 8 月，Microsoft 发布了用于 Visual Studio 和.NET Framework 的 Service Pack 1，其中引入了一些重要的新功能，如 ADO.NET Entity Framework 和动态数据。

目前该软件的最新版本是 Visual Studio 2010(通常读作 twenty-ten)和 ASP.NET 4.0，它是在已成功发行的 Visual Studio 2008 和 ASP.NET 3.5 的基础之上构建的，其中保留了很多令人喜爱的功能，并增加了一些其他领域的新功能和工具。



1.2.2　ASP.NET 的开发环境

进行 ASP.NET 开发时需要使用的语言是 Visual Basic.NET 或者 C#，这两种语言都是.NET 环境下的程序设计语言，但并不是必须使用.NET 集成开发环境才能进行 ASP.NET Web 程序设计。因为 ASP.NET 文件实际上是一个纯文本的文件，编译工作是在用户向服务器第一次发出对该文件的 HTTP 请求时由 Web 服务器进行的，并不是由 VS 完成的。从理论上讲，用记事本或其他文本编辑器就可以编写 ASP.NET Web 应用程序，但大多数开发人员还是希望安装 Microsoft Visual Web Developer 2010(VWD)。VWD 是专门为构建 ASP.NET Web 站点而开发的，其中包含了大量有助于快速创建复杂 ASP.NET Web 应用程序的工具。

Visual Web Developer 有两个版本：一个是独立而免费的版本，称为 Microsoft Visual Web Developer 2010 Express；还有一个版本是作为较大的开发套件 Visual Studio 2010 的一部分，它有不同的版本可用，且各个版本的价格各不相同。虽然 VWD 的 Express 版本是免费的，但是它包含了创建复杂且功能丰富的 Web 应用程序所需的所有功能和工具。本书中的所有示例都可以用该版本构建出来。





1. 获取 Visual Web Developer 2010

可以从 Microsoft 站点 http://www.microsoft.com/express/上下载 VWD 的免费版本。在 Express 的主页上，依次单击 Download 链接，直到打开提供了下载 Express 产品的下载页面，其中包括 Visual Web Developer 2010 Express 版本。在这个页面上可以以 Web 安装方式下载 Visual Web Developer 2010 Express 版本，这里只下载安装程序，文件的其余部分将在安装过程中下载。



提示

也可以从这个下载页面上以 ISO 映像方式下载 VWD 2010 Express 产品，以便刻录到 DVD 上进行安装。

读者还可以从 www.microsoft.com/web 和 www.asp.net/vwd/站点上下载 Microsoft Web Platform Installer(WPI)应用程序，其中就包含了 VWD。除了 VWD 以外，这个工具还便于访问其他许多与 Web 开发相关的工具和程序。通过使用 WPI 这个优秀的工具，可以同时获得大量与 Web 开发相关的程序和工具。

在本书中，以 C#作为编程语言，以 VWD 2010 作为开发环境，进行 ASP.NET 动态网站开发，因此需要把它安装到开发机器上。

2. 安装 VWD Express 版本

Visual Web Developer 的安装很简单，只是过程有点长。根据所选的安装方法、计算机配置和 Internet 连接速度，安装 VWD 可能需要半个小时到一个小时，甚至更长时间。

安装 Visual Studio 2010 的完整版本与之相似，只是中间步骤可能略有不同。不管安装 VWD 的哪个版本，都要安装 SQL Server 2008 Express Edition 的 Service Pack 1 版—— 本书的很多示例都会用到这个组件。如果安装的是 Visual Studio 2010 的完整版，那么在安装过程中会看到要安

装的功能列表中包括安装 SQL Server 的选项。如果安装 VWD Express 版本，Installer Options 对话框中就会出现选择 SQL Server 的选项。Web Platform Installer 也包括相似的选项，允许安装 SQL Server 2008 Express 的 SP1 版或者稍后定位到 Web Platform | Database 下。

(1) 运行从 Microsoft Web 站点上下载的文件，或者从安装光盘上启动安装文件，将开始下载 Web 平台安装程序文件，如图 1-3 所示。

图 1-3　下载安装程序文件

(2) 下载完以后，打开【Web 平台安装程序 3.0】，如图 1-4 所示。



图 1-4　启动 Web 平台安装程序 3.0

(3) 单击【安装】按钮，出现软件的许可条款，如图 1-5 所示。在该列表中，包括 VWD 2010 依赖的所有应用程序和组件。如果没有看到 SQL Server 选项，则表示已经安装过了。



图 1-5　软件许可条款



(4) 阅读并接受许可条款，单击【我接受】按钮将开始下载安装，下载和安装的过程与需要安装的组件有关。

(5) 完成了应用程序的安装后，可能会出现一个对话框，要求重启计算机。重启后，VWD 就可以使用了。

提示

在安装 VWD 2010 Express 版本中，包括了 SQL Server 2008 Express R2，它是 Microsoft 数据库引擎的免费版本，本书后面涉及数据库的 Web 应用程序开发都是使用该版本。

3. VWD 2010 提供的功能

VWD 2010 提供了下列功能。

- 网页设计：VWD 2010 内置功能强大的网页编辑器，包含所见即所得的编辑模式和 HTML 编辑模式，以及智能感应功能和验证功能，支持所见即所得的拖拽界面，可以创建美观、易用的网站。
- 网页设计功能：VWD 2010 支持页面模式，使用主题和外观保持一致的页面外观，从而统一管理网页的排版与布局。
- 代码编辑：VWD 2010 提供代码编辑器，使用户可以使用 Visual Basic .NET 或 C#编写动态网页的代码。代码编辑器拥有语法修饰和智能感应功能。
- 调试：提供调试器，帮助用户查找程序中的错误。
- 控件：ASP.NET Web 服务器控件整合了创建网站所需的大部分功能，用户可以快速开发 Web 应用程序。
- 数据访问：支持用户在网页中显示和编辑数据。数据可以位于各种数据存储区中，其中包括数据库或 XML 文件。在很多情况下，用户无需编写任何代码，即可向网页中添加和编辑数据。
- 安全性、个性化设置：提供内置的应用程序服务，用户可以向网站中添加用于确保登录安全性的成员资格；提供配置文件属性，维护用户特定的信息；另外还提供了其他功能，其中的大部分功能都不要求编写任何代码。
- 对文件传输协议(FTP)的内置支持：使用 VWD 2010 的 FTP 功能，可以直接连接到服务器，然后在该服务器上创建和编辑文件。
- 内置 Web 服务器：VWD 2010 包含了一个内置的 Web 服务器，方便开发人员创建和调试 ASP.NET Web 应用程序。因此，用户不需要再安装和配置 IIS 服务器，就可以开发 ASP.NET Web 应用程序。
- 微软 AJAX：微软 AJAX 的一个重要特性是，它与其他客户端架构(包括 jQuery)具有很好的互操作性。除了实现无闪烁页面的控件之外，微软 AJAX 还提供了更多的服务器控件来创建交互式的且有响应的用户界面。
- JQuery 1.4：jQuery 库的主要关注点一直是简化访问 Web 页面元素的方法、帮助处理客户端事件、提供视觉效果(如动画)支持，以及使得在应用程序中使用 AJAX 变得更加简单。VWD 2010 包含了目前最新的稳定版本 jQuery 1.4。使用 ASP.NET Web 站点模板

创建的任意新 Web 站点都包含一个 Scripts 文件夹，其中已经包含了必要的 jQuery 文件。

- MVC 2.0：ASP.NET MVC 模式是一种表现模式。它将 Web 应用程序分成 3 个主要组件，即模型(Model)、视图(View)、控制器(Controller)。在 ASP.NET MVC 中，“请求——处理——响应”的模型变得更加简单。View 和 Controller 之间不再有强耦合，而且页面没有复杂的生命周期。
- 多显示器支持：比如将代码编辑器放置在主显示器中，将输出窗口、类图窗口、代码定义窗口等提供辅助信息的窗口放置在副显示器中，这样就可以在主窗口中编辑代码，同时有需要的时候，可以及时地从辅助窗口中得到一些有用的辅助信息。

1.3 使用 VWD 创建 Web 应用程序

现在已经安装了 VWD，接下来就可以启动并使用它来创建 ASP.NET 应用程序了。本节将介绍 VWD 2010 的 IDE 集成环境，以及用 VWD 2010 创建第一个 ASP.NET 应用程序。

1.3.1 VWD 2010 IDE 环境介绍

单击 Windows 的【开始】菜单，选择【所有程序】|【Microsoft Visual Studio 2010 Express】|【Microsoft Visual Web Developer 2010 Express】命令，启动 VWD 2010，如图 1-6 所示。

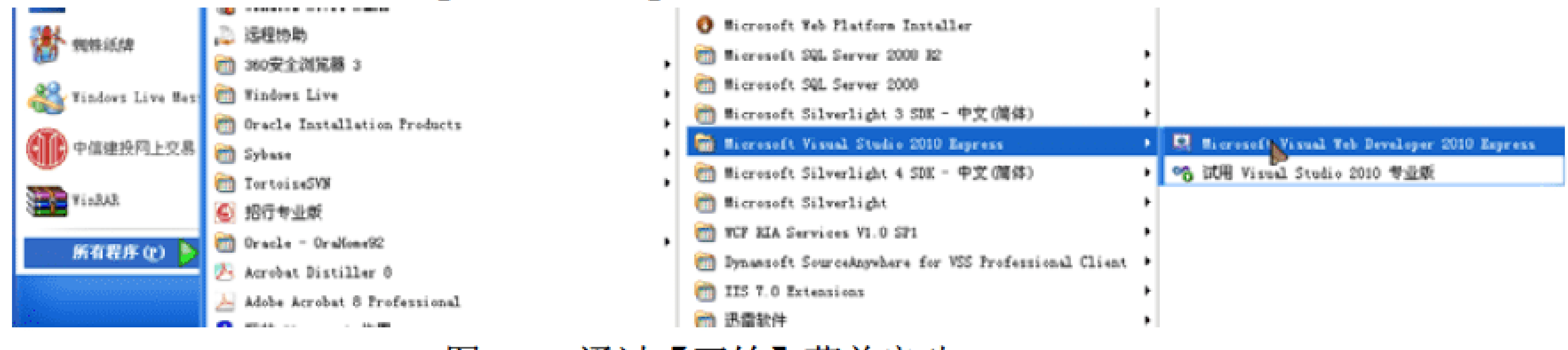


图 1-6 通过【开始】菜单启动 VWD 2010

首先出现的是软件版本的界面，如图 1-7 所示。几秒钟之后，软件版本的界面消失，出现【起始页】界面，如图 1-8 所示。



图 1-7 软件版本的界面

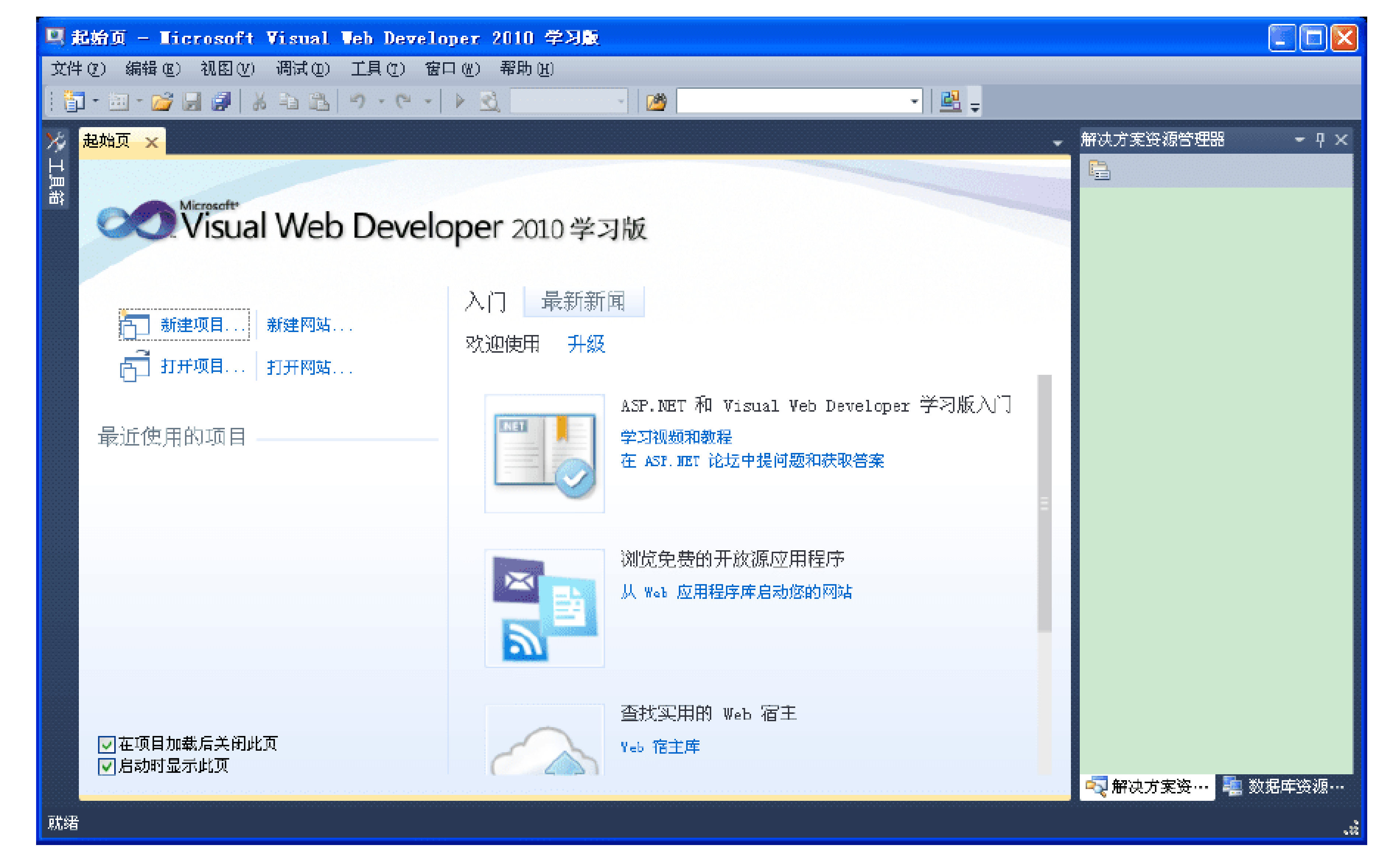



图 1-8　【起始页】界面

【起始页】页面包括【最近使用的项目】和联机资源以及新建和打开项目的快捷操作。为了介绍 VWD 2010 的操作环境，这里先新建一个网站，在【起始页】页面中单击【新建网站】链接，或者选择【文件】|【新建网站】命令，打开【新建网站】对话框，如图 1-9 所示。



图 1-9　【新建网站】对话框

该对话框显示了已经安装的模板，这里选择【ASP.NET 网站】模板，在【Web 位置】下拉列表中选择【文件系统】选项，然后在后面的文本框中输入存储位置，或者单击【浏览】按钮选择一个新位置，然后单击【确定】按钮即可创建一个 ASP.NET 网站，其中包括一个名为 Default.aspx

的标准页面、一个 Web.config 文件以及一个空的 App_Data 文件夹，如图 1-10 所示。

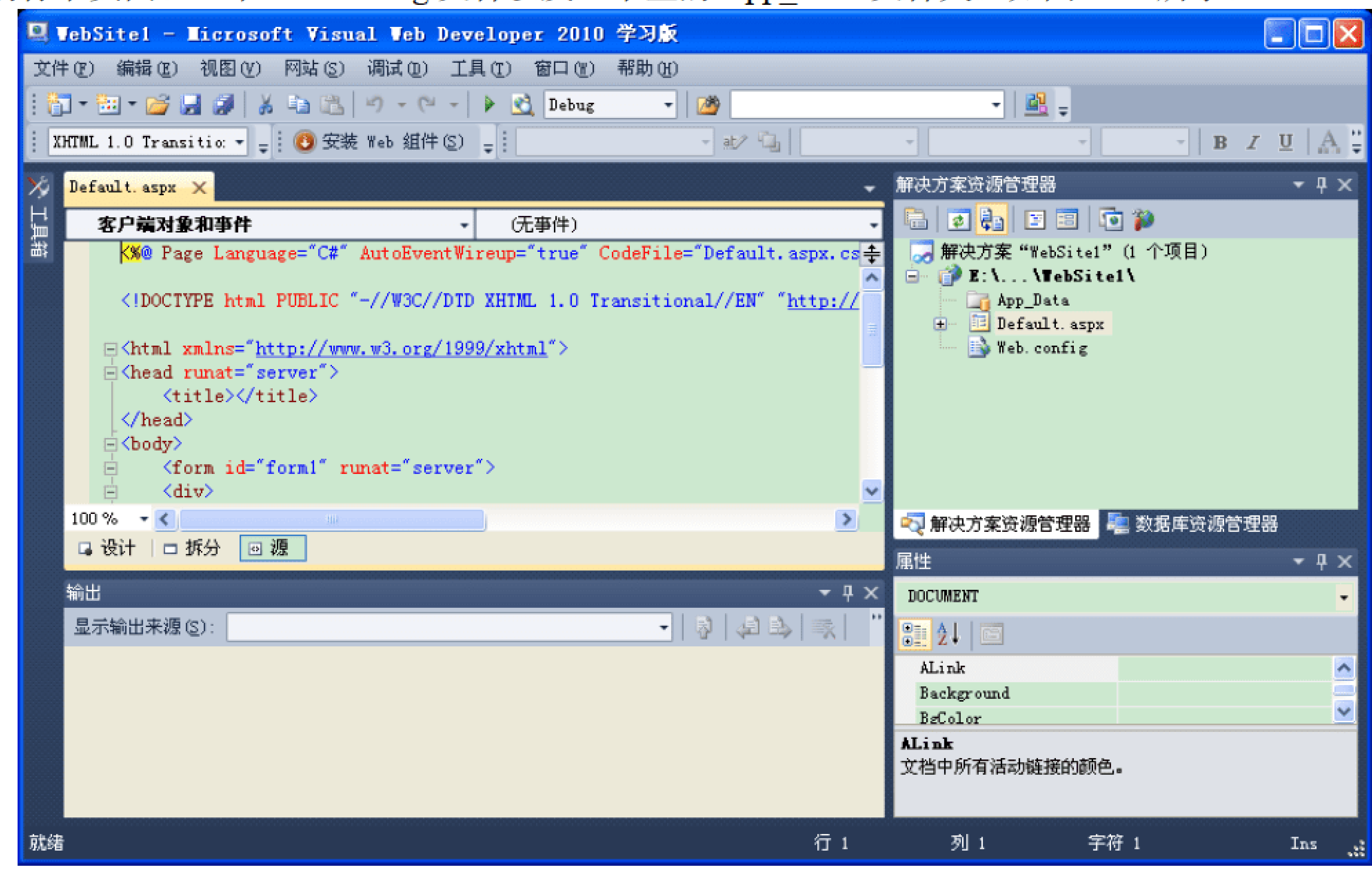


图 1-10　VWD 2010 主开发界面

VWD 2010 的主界面包括标题栏、菜单栏、工具栏、工具箱、解决方案资源管理器、数据库资源管理器、属性窗口、文档窗口等。

1. 菜单栏

开发界面的最上方是标题栏，标题栏的下面就是菜单栏，包括【文件】、【编辑】、【视图】、【网站】、【调试】、【工具】、【窗口】和【帮助】8 个主菜单。根据执行的具体任务不同，主菜单也会有很大的变化，因此，在使用应用程序的过程中就会发现某些菜单有时出现、有时消失。

2. 工具栏

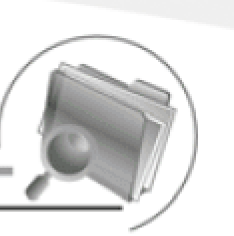
菜单栏的下面就是工具栏。利用不同的工具栏，可以快速地访问 VWD 中的大部分常用功能。图 1-10 中只显示了 4 个工具栏(HTML 源编辑、Web 平台安装程序、标准和格式设置)，如果要打开或关闭某个工具栏，可以右击现有的工具栏，或者选择【视图】|【工具栏】菜单，从弹出的子菜单中选择相应的菜单项即可，如图 1-11 所示。

3. 工具箱

默认情况下，在主窗口的左侧，可以看到折叠的工具箱选项卡，将鼠标指针移动到该选项卡上悬停几秒，工具箱就会展开，如图 1-12 所示。

与菜单栏和工具栏一样，在执行不同的任务时，工具箱也可能会变化，以显示相关的控件。可以简单地通过鼠标拖动将工具箱中的控件拖放到页面中的合适位置。





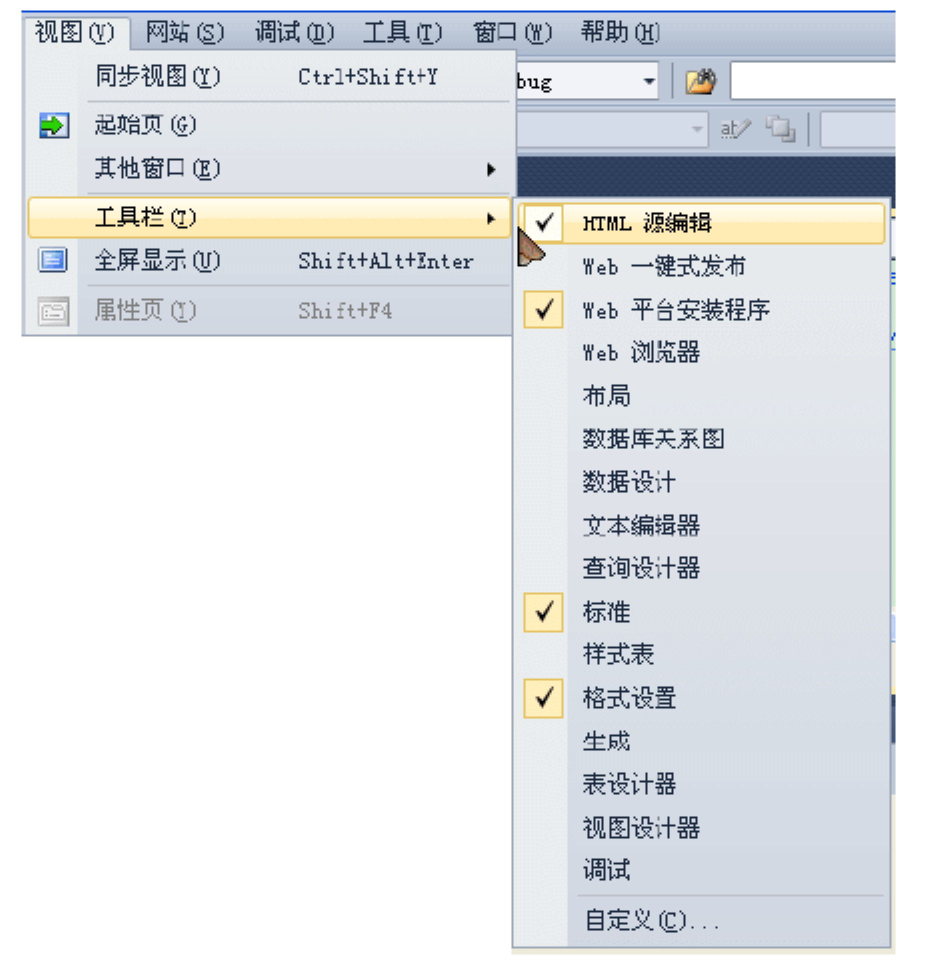


图 1-11　【视图】|【工具栏】子菜单

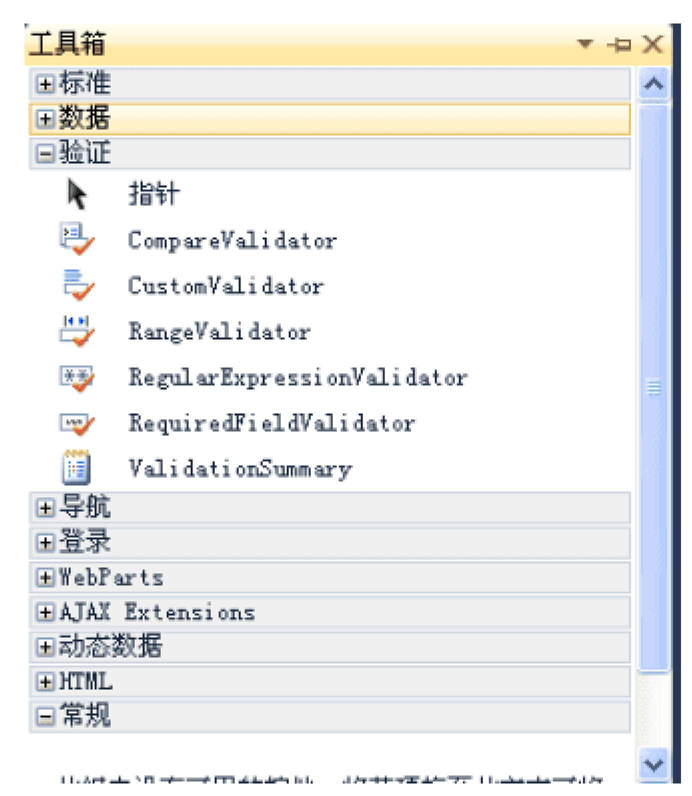


图 1-12　工具箱

工具箱中的控件包含多个分类，用户可以根据需要展开或折叠某个分类，以便找到需要的控件。如果在主窗口左侧找不到工具箱，可以按【Ctrl+Alt+X】组合键或者选择【视图】|【其他窗口】|【工具箱】命令来打开它。

4. 解决方案资源管理器

窗口的右上角是【解决方案资源管理器】面板，如图 1-13 所示。在该面板中，文件被分门别类地存储在不同的文件夹中，可以通过该面板向站点中添加新的文件夹和文件，从项目中删除文件，更改文件或文件名等。解决方案资源管理器的大部分功能都集中在它的右键快捷菜单中。

【解决方案资源管理器】面板一般与【数据库资源管理器】面板集成在一起显示，通过该面板，可以使用数据库，创建新数据库和打开现有数据库，以及向数据库中添加新的表和查询工具。

5. 【属性】面板

【属性】面板位于窗口的右下角，如图 1-14 所示。通过该面板可以查看和编辑项目、文件、控件、页面本身的属性以及其他内容。

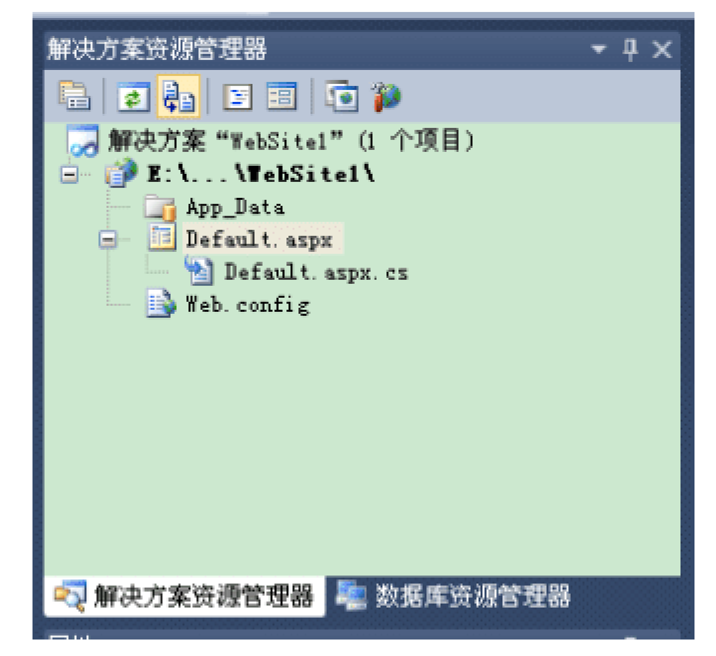


图 1-13　【解决方案资源管理器】面板

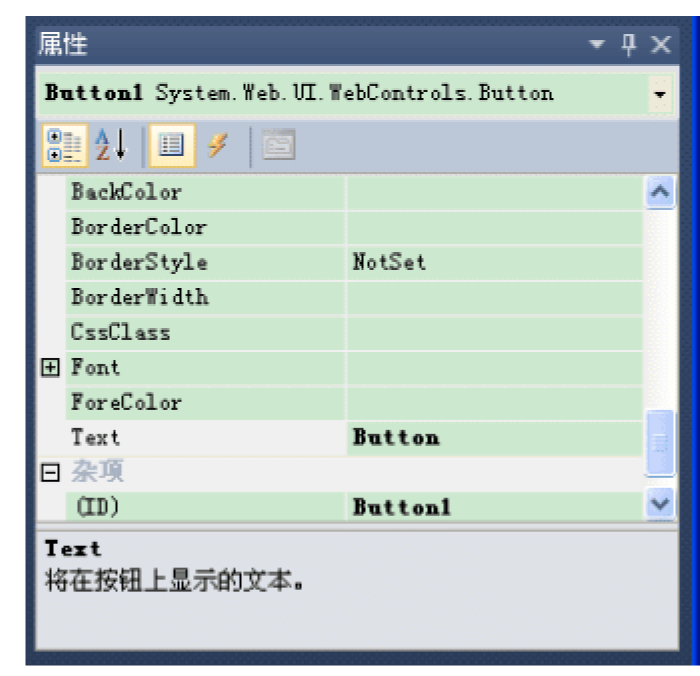


图 1-14　【属性】面板



6. 文档编辑区

文档编辑区是界面的主要区域，大部分动作都是在这里发生的。在文档编辑区的下方，有 3 个视图按钮，分别是【设计】、【拆分】和【源】按钮。在操作含有标记的文件(如 aspx 和 html 文件)时，这些按钮会自动出现。单击【设计】按钮可以打开页面的设计视图，在这里可以看到页面在浏览器中的效果；单击【源】按钮将打开源视图，在此可以看到页面的源代码；单击【拆分】按钮可以同时打开设计视图和源视图。

默认情况下，文档编辑区是一个带选项卡的区域，各文件通过选项卡呈现，文件名在编辑区顶部。如果选项卡上的文件名带有“*”，则说明该文件的内容修改过但还没有保存。

7. 其他面板

除了上面介绍的几个组成部件之外，VWD 2010 还有很多工具面板，包括输出、错误列表、书签、查询结果等面板。这些面板都可以通过【视图】菜单下面的相应命令打开。

1.3.2 第一个 Web 应用程序

本节将通过 VWD 2010 创建第一个 ASP.NET Web 应用程序。通过这个例子，读者将可学会如何通过 VWD 2010 创建 Web 应用程序，并对在浏览器中浏览 ASP.NET 页面时后台的工作有一个很好的了解。

1. 新建网站

【例 1-1】通过 VWD 2010 新建网站，了解通过 VWD 2010 开发 Web 应用程序的步骤。

(1) 通过【开始】菜单启动 VWD 2010，选择【文件】|【新建网站】命令，打开【新建网站】对话框，新建网站【例 1-1】。

(2) 通常情况下，VWD 会为新建网站创建一个新的子目录。新建的网站包括一个名为 Default.aspx 的页面，该页面是网站的默认主页面和入口点；另外还包括一个名为 Default.aspx.cs 的代码文件，该文件是 Default.aspx 页面的后台代码；除此之外，还有一个名为 Web.config 的配置文件。

(3) 修改 Defualt.aspx 页面的代码如下所示：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>我的第一个 ASP.NET 站点</title>
</head>
<body>
```



```
        <h2>欢迎进入 ASP.NET 4 的世界</h2>
        <form id="form1" runat="server">
        <div>
        欢迎使用 VWD 2010，现在时间是：<% =System.DateTime.Now %>
        </div>
        </form>
    </body>
    </html>
```

(4) 选择【调试】|【启动调试】命令，如图 1-15 所示，或者按【F5】键，或单击工具栏中的 按钮，将编译并生成网站，同时启动调试。

(5) 如果代码输入正常，主窗口下方的【输出】窗口中将出现生成成功的信息，如图 1-16 所示。如果有语法错误，则在【错误列表】中将逐一列出所有错误，双击某项错误将跳转到相应的代码处。



图 1-15　选择【启动调试】命令

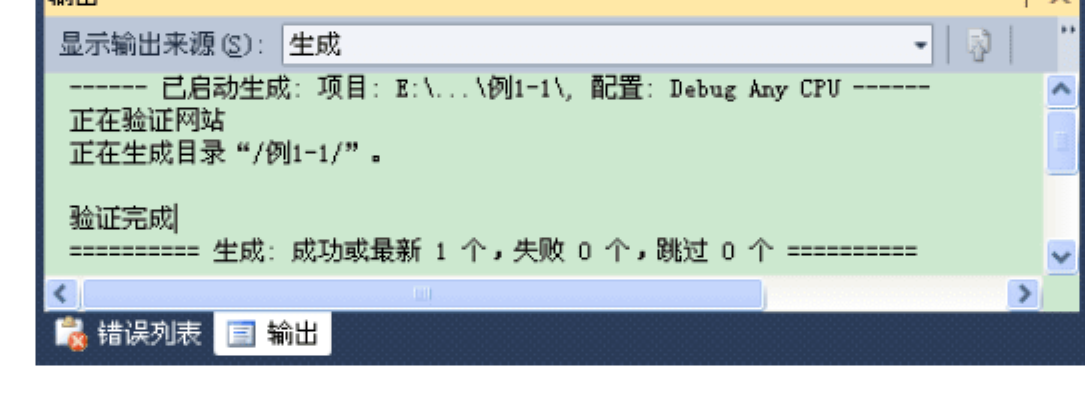


图 1-16　【输出】窗口

(6) 此时将弹出【未启用调试】对话框，如图 1-17 所示，如果选中【修改 Web.config 文件以启用调试】单选按钮，则以后启动此工程时将不再弹出该对话框，而默认启动调试；如果选中【不进行调试直接运行】单选按钮，则不启动调试，等同于用户按【Ctrl+F5】组合键。

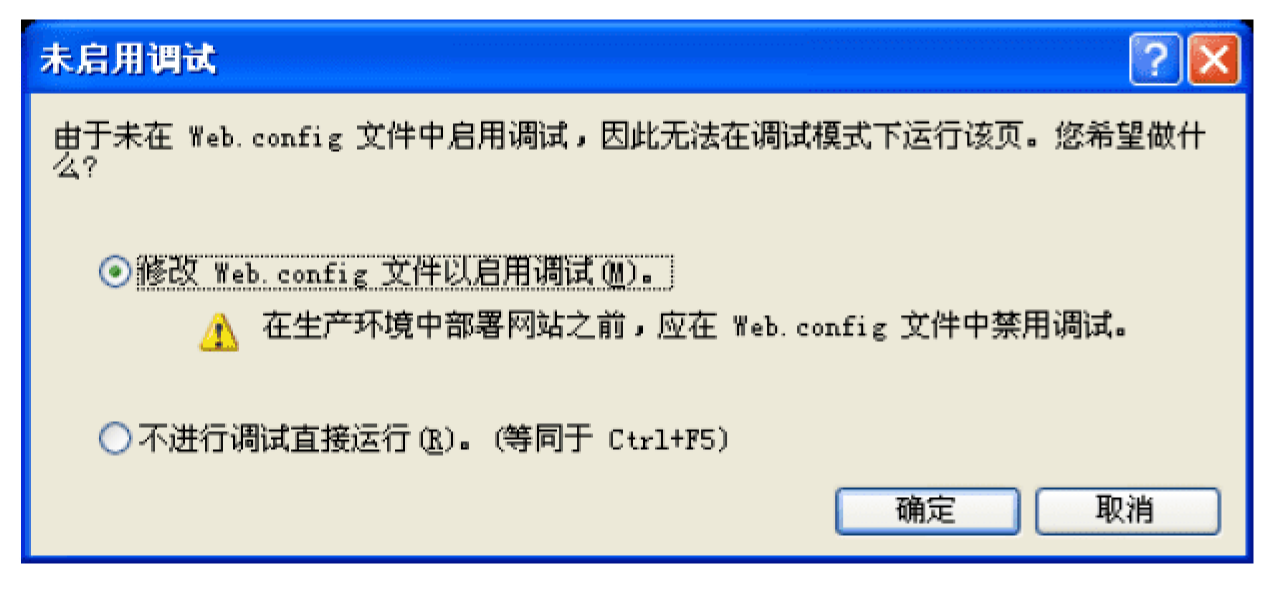


图 1-17　【未启用调试】对话框

(7) 单击【确定】按钮后，将自动启动默认的 Web 浏览器，同时打开该页面，如图 1-18 所示。





图 1-18　页面运行效果



提示

如果在页面中没有看到时间和日期，或者收到了错误消息，可查看一下欢迎消息中的代码，确定是否以前尖括号(<)开头，后面跟着一个百分号和一个等号，以一个百分号和另一个后尖括号(>)结束。尽管如此，还是要确保输入和此处完全相同的代码，包括大小写也要一致。

(8) 此时，在 Windows 的任务栏中会出现一个带屏幕提示的小图标，这个图标属于 ASP.NET Development Server。该 Web 服务器由 VWD 自动启动，以响应客户端对页面的请求。双击该图标将打开如图 1-19 所示的详细信息。



图 1-19　ASP.NET Development Server

2. 工作原理

虽然本例中创建的 Web 站点相当简单，但是让 Default.aspx 页面显示在浏览器中的过程却没有那么简单。ASP.NET 页面(根据它的扩展名，也称为 ASPX 页面)本身并不能做太多的事。在浏览器能够显示它之前，需要一个 Web 服务器对它进行处理，这就是 VWD 自动启动内置的 ASP.NET Development Server 来处理页面请求的原因。接下来，它会启动默认的 Web 浏览器并定向到本例中的 Web 服务器地址：http://localhost:1816/%E4%BE%8B1-1/Default.aspx，需要注意的



是：每次启动 Web 服务器时这个地址中的端口号可能会不同，因为该端口是 VWD 随机选择的。

当在 VWD 中创建一个页面时，就向它添加了一个标记(markup)。ASPX 页面中的标记由以下内容组成：纯文本、HTML、ASP.NET 服务器控件的代码、用 Visual Basic .NET 或 C#编写的代码等。

当客户端在浏览器中请求一个 ASPX 页面时，Web 服务器就会处理该页面，执行它在文件中找到的所有代码，并有效地将 ASP.NET 标记转换为纯 HTML，然后发送回客户端浏览器。

要查看最终的 HTML 与原始的 ASPX 页面的区别，可以在浏览器中打开该页面的源代码。选择【查看】|【源文件】命令，将会打开一个默认的文本编辑器，用于显示该页面的 HTML 代码。其中的大部分 HTML 与原始的 ASPX 页面相似。然而，如果看一下显示欢迎消息和当前日期与时间的那一行代码，就会知道它们有很大的区别。现在看到的不是尖括号和百分号之间的代码，而是实际的日期和时间。如下所示：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
我的第一个 ASP.NET 站点
</title></head>
<body>
    <h2>欢迎进入 ASP.NET 4 的世界</h2>
    <form method="post" action="Default.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUJOTU4MjMyMzI1ZGRyJQkrhRQxLtgmJwQTfVUVAs+BRrzD2CE4smjwDM7znw==" />
</div>
    <div>
    欢迎使用 VWD 2010，现在时间是：2011-5-27 20:18:56
    </div>
    </form>
</body>
</html>
```



1.3.3　初识 ASP.NET 标记

在某种程度上，ASP.NET 服务器控件的标记与 HTML 的标记很相似。它也像 HTML 一样用尖括号和结束标记表示标记、元素和特性。然而，它们之间还是有一些不同之处的。

对于初学者来说，大部分 ASP.NET 标记都是以“asp:”前缀开头。例如，ASP.NET 中的按钮类标记如下：

```
<asp:Button ID="Button1" runat="server" Text="提交" />
```



提示

此标记是用斜线字符(/)自闭合的，所以不必另外输入结束标记。

另外，这里的标记和属性名不必都是小写。因为服务器上有一个 ASP.NET 服务器控件，所以不必在客户端的浏览器中使用 XHTML 规则。然而，当要求服务器控件将它的 HTML 发送到配置为输出 XHTML 的页面时，它就会应用 XHTML 规则。因此，当在浏览器中呈现为 XHTML 时，这个按钮的代码如下：

```
<input type="submit" name="Button1" value="提交" id="Button1" />
```

将服务器控件转换为它的 HTML 表示的过程与前面显示当前日期的过程相似。服务器控件由 ASP.NET 处理程序在服务器中进行处理，并将生成的 HTML 发送到浏览器中显示。



1.4 上机练习

本章的上机练习主要熟悉创建 ASP.NET 4.0 应用程序的过程，进一步了解其工作原理。

本练习通过服务器控件显示欢迎信息并显示当前日期和时间，与前面的例 1-1 效果类似，请读者比较两者的区别。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 1】，单击【确定】按钮。

(2) 切换到 Default.axpx 页面的【设计】视图，从工具箱中拖动一个 Label 控件到页面中，如图 1-20 所示。

(3) 在页面的空白处双击鼠标，将自动添加页面的 Load 事件，并跳转到后台代码文件 Default.apsx.cs 中，在此添加如下代码:

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "欢迎进入 ASP.NET 4 的世界，现在时间是：" + DateTime.Now;
}
```

(4) 启动调试，在默认浏览器中打开该页面，如图 1-21 所示。

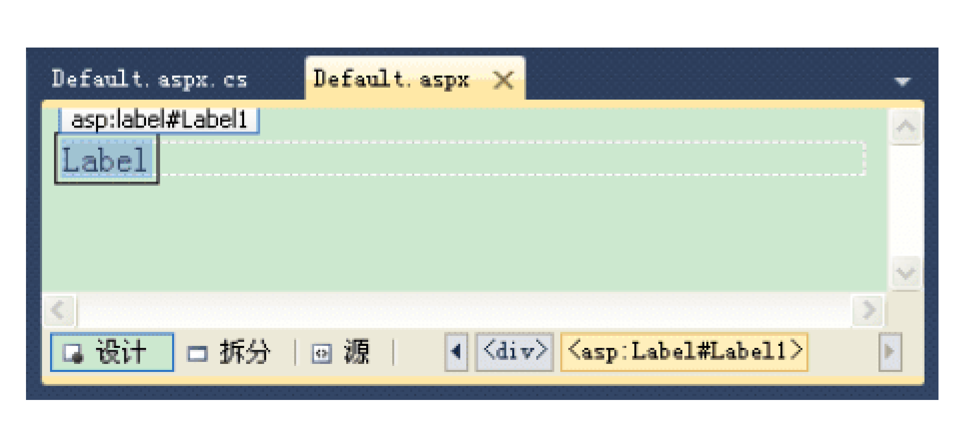
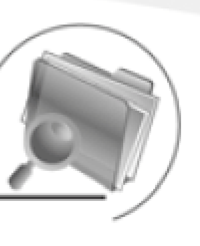


图 1-20　在页面中添加 Label 控件



图 1-21　页面运行效果



1.5　习题

1. 参照本章内容，获取 VWD 2010 并安装。
2. 简述 ASP.NET 的工作原理。
3. 参照本章上机练习，新建网站，通过 Label 控件显示当前时间。





ASP.NET 基础知识

学习目标

本章主要介绍 ASP.NET 的一些基础知识，学习和掌握这些知识是以后进行 ASP.NET 程序开发的基础和前提。这些知识主要包括 ASP.NET 的页面框架和页面类，ASP.NET 的内置对象以及 ASP.NET 的配置文件 Web.config 和全局文件 Global.asax。本章是学习 ASP.NET 非常关键的一章，是从认识了解到使用 ASP.NET 的一个关键点。

本章重点

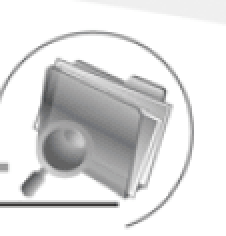
- 了解 ASP.NET 的文件类型
- 理解 ASP.NET 页面的运行机制和页面的生命周期
- 掌握创建和使用常用内置对象的方法
- 掌握 Cookie 的使用以及设置和检索 Cookie 的方法
- 掌握 ASP.NET 的配置管理

2.1 ASP.NET 应用程序基础

ASP.NET 应用程序与传统的桌面型应用程序不同。传统的桌面型应用程序是一个独立的 exe 文件；而 ASP.NET 应用程序则总是被分成若干个 Web 页面，这样，用户就可以从不同的入口进入一个 ASP.NET 应用程序，或者跟随超链接从一个 Web 应用程序导航到另一个 Web 应用程序。

每一个 ASP.NET 应用程序都共享一组资源和配置设置，另一个 ASP.NET 应用程序则不能共享这些资源和配置，即使它们位于同一个 Web 服务器上。从技术的角度来讲，每一个 ASP.NET 应用程序都在一个独立的“应用程序域”(Application Domain)中执行。应用程序域是内存中相互隔离的内存区域，这使得当一个 Web 应用程序出现故障时，不会影响到另一个 Web 应用程序。

标准的 ASP.NET 应用程序定义为 Web 服务器上的一系列文件、页面、处理程序、功能模块



和可执行代码的集合，这些可执行文件可以从当前网站的虚拟目录及其子目录中进行调用。换句话说，虚拟目录就是界定 Web 应用程序的基本组织结构。如图 2-1 所示是一个 Web 服务器的应用程序域的结构，其中包含了两个独立的 Web 应用程序。

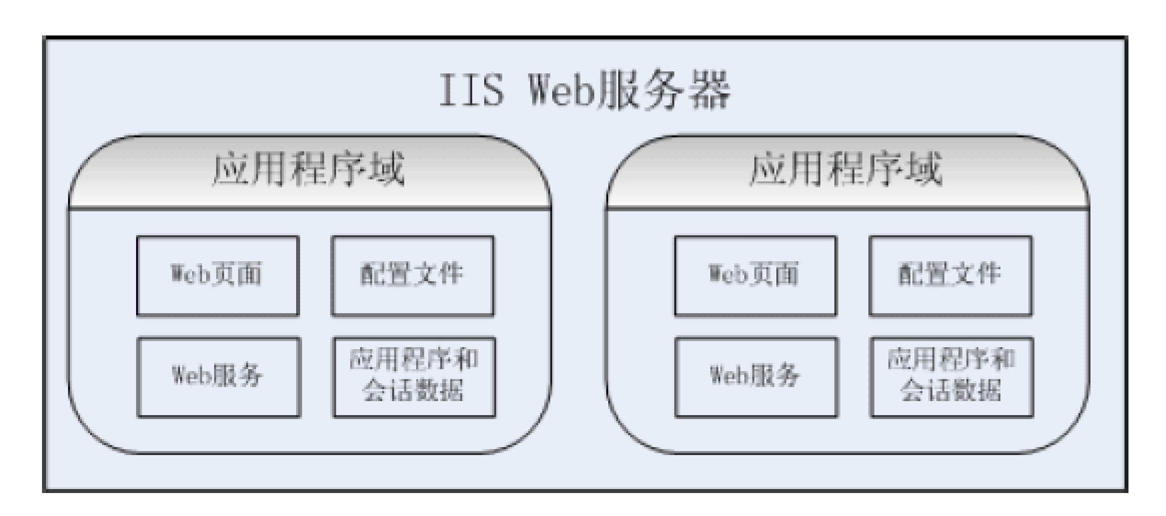
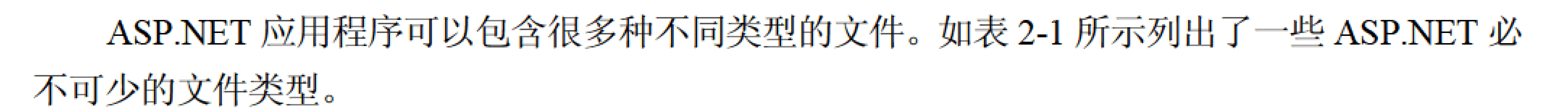


图 2-1　ASP.NET 应用程序域

2.1.1　ASP.NET 的文件类型





ASP.NET 应用程序可以包含很多种不同类型的文件。如表 2-1 所示列出了一些 ASP.NET 必不可少的文件类型。



表 2-1　ASP.NET 文件类型

文 件 类 型	描　述
.aspx 文件	.aspx 文件是 ASP.NET 的 Web 页面文件，与传统 ASP 应用程序中的.asp 文件对应。.aspx 文件中包含页面的用户界面，也可以包含一些基本的应用程序代码。用户向 Web 服务器请求某个.aspx 文件，或者导航到某个.aspx 文件时，将进入一个 Web 应用程序
.ascx 文件	.ascx 文件是 ASP.NET 的用户控件。用户控件与 Web 页面类似，但用户无法直接访问.ascx 文件。实际上，.ascx 文件必须嵌入到一个 ASP.NET 页面中才能运行。可以使用用户控件来开发一些用户界面的模块，并在其他页面中直接使用这些用户控件，而不必重复编写相同的代码
.asmx 文件	.asmx 文件是 ASP.NET Web 服务文件。Web 服务是一组方法的集合，这些方法可以通过 Internet 进行调用。Web 服务的工作原理与 Web 页面完全不同，但是 Web 服务与 Web 页面共享同一应用程序中的各种资源、配置设置和内存区域
Web.config 文件	Web.config 文件是一个基于 XML 的 ASP.NET 配置文件。该配置文件包含了自定义的安全设置、状态管理、内存管理等多种配置的设置。本书中很多地方都会用到该文件的配置
Global.asax 文件	Global.asax 文件是一个全局应用程序文件。在该文件中，可以定义全局变量(全局变量可以在当前 Web 应用程序中的所有页面中访问)，还可以定义应用程序的全局事件(如当一个 Web 应用程序启动时的事件)
.cs 文件	.cs 文件是 Web 页面的后台代码文件，其中包含执行页面逻辑功能的 C#代码。.cs 文件将应用程序的逻辑从 Web 页面的用户界面中分离出来

除了上表所列的文件类型以外，Web 应用程序还可以包含一些其他的资源文件，这些资源文件并不是专门用于 ASP.NET 的。例如，在虚拟目录中，可以包含图片文件、HTML 文件或 CSS 样式文件。这些资源文件可以在某个 ASP.NET 页面中使用，也可以独立使用。

2.1.2　ASP.NET 应用程序的目录结构

每个 Web 应用程序都应该具有一个良好的目录结构规划。开发者在对程序进行设计时应该将特定类型的文件存放在某些文件夹中，以方便今后开发中的管理和操作。ASP.NET 保留了一些特殊的子目录，程序开发人员可以直接使用，并且还可以在应用程序中增加任意多个文件和文件夹。

提示

在一个典型的 ASP.NET 应用程序中，并不一定需要使用全部的特殊子目录。

1. Bin 子目录

Bin 子目录包含应用程序所需的用于控件、组件或者需要引用的任何其他代码的可部署程序集。该目录中存在的任何.dll 文件将自动地链接到应用程序。可以在 Bin 子目录中存储编译的程序集，Web 应用程序中其他任意处的代码会自动引用该目录。典型的实例是：如果为自定义类编译好了代码，那么就可以将编译后的程序集复制到 Web 应用程序的 Bin 子目录中，这样，所有页都可以使用这个类了。

Bin 子目录中的程序集无需注册。只要.dll 文件存在于 Bin 子目录中，ASP.NET 就可以识别它。如果更改了.dll 文件，并将它的新版本写入到了 Bin 子目录中，则 ASP.NET 会检测到更新，并对随后的新页请求使用新版本的.dll 文件。

将编译后的程序集放入 Bin 子目录中会带来安全风险。如果是自己编写和编译的代码，那么设计者是了解代码的功能的。但是，设计者必须像对待任何可执行代码一样来对待 Bin 子目录中已编译的代码。在完成代码测试并确信已了解代码功能之前，要对已编译的代码保持谨慎的态度。Bin 子目录中的程序集的作用范围为当前应用程序，因此，它们无法访问当前 Web 应用程序之外的资源或调用当前 Web 应用程序之外的代码。另外，在运行时，程序集的访问级别是由本地计算机上指定的信任级别决定的。

2. App_Code 子目录

App_Code 子目录在 Web 应用程序根目录下，它存储所有应当作为应用程序的一部分动态编译的类文件。这些类文件自动链接到应用程序，而不需要在页面中添加任何显式指令或声明来创建依赖性。App_Code 目录中放置的类文件可以包含任何可识别的 ASP.NET 组件——自定义控件、辅助类、build 提供程序、业务类、自定义提供程序和 HTTP 处理程序等。



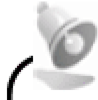
在开发时，对 App_Code 子目录的更改会导致整个应用程序的重新编译。对于大型项目，这可能不受欢迎，而且很耗时。为此，鼓励大家将代码进行模块化处理到不同的类库中，按逻辑上相关的类集合进行组织。应用程序专用的辅助类大多应当放置在 App_Code 文件夹中。

App_Code 子目录中存放的所有类文件应当使用相同的语言。如果类文件使用两种或多种语言编写，则必须创建特定语言的子目录，以包含用各种语言编写的类。一旦根据语言组织这些类文件，就要在 Web.config 文件中为每个子目录添加设置，有关 Web.config 文件的使用将在后面进行详细介绍。

App_Code 子目录和 Bin 子目录是 ASP.NET 网站中的共享代码文件夹，如果 Web 应用程序要在多个页之间共享代码，就可以将代码保存在 Web 应用程序根目录下的这两个特殊目录中。当创建这些子目录并在其中存储特定类型的文件时，ASP.NET 将使用特殊方式进行处理。

3. App_Data 子目录

App_Data 子目录保存应用程序使用的数据库。它是一个集中存储应用程序所用数据库的地方，是 ASP.NET 为程序提供存储自身数据的默认位置，该文件夹内容不由 ASP.NET 处理。它通常以文件(诸如 Microsoft Access 或 Microsoft SQL Server 数据库、XML 文件、文本文件以及应用程序支持的任何其他文件)的形式对数据进行存储。当然，也可以将数据文件保存到其他目录中。



提示

默认情况下，ASP.NET 帐户被授予对该子目录的完全访问权限。如果要改变 ASP.NET 帐户，一定要确保新帐户被授予对该目录的读/写访问权。

4. App_GlobalResources 子目录

App_GlobalResources 子目录用于保存 Web 应用程序中的全局资源文件。资源文件是一些字符串表，当应用程序需要根据某些事情进行修改时，资源文件可用于这些应用程序的数据字典。可以在 App_GlobalResources 子目录中添加程序集资源文件(.resx)，它们会动态编译，成为解决方案的一部分，供程序中的所有.aspx 页面使用。在使用 ASP.NET 1.0/1.1 时，必须使用 resgen.exe 工具把资源文件编译为.dll 或.exe，才能在解决方案中使用。而从 ASP.NET 3.5 开始，资源文件的处理就容易多了，除了字符串之外，还可以在资源文件中添加图像和其他文件。

当需要开发一个支持多种语言的 Web 网站时，该目录用于进行本地化设置。

5. App_LocalResources 子目录

App_GlobalResources 子目录用于合并可以在应用程序范围内使用的资源。如果对构造应用程序范围内的资源不感兴趣，而对只能用于一个.aspx 页面的资源感兴趣，就可以使用 App_LocalResources 子目录。可以把专用于页面的资源文件添加到该目录中，方法是构建.resx 文件名，如下所示：

```
Default.aspx.resx
Default.aspx.fi.resx
```

```
Default.aspx.ja.resx
Default.aspx.en-gb.resx
```

现在就可以从 App_LocalResources 目录的相应文件中检索在 Default.aspx 页面上使用的资源声明了。如果没有找到匹配的资源，就默认使用 Default.aspx.resx 资源文件。

6. App_WebReferences 子目录

App_WebReferences 子目录用于保存当前 Web 应用程序中用到的 Web 服务引用。

7. App_Themes 子目录

App_Themes 子目录用于存放 Web 应用程序中使用的主题。主题是为站点上的每个页面提供统一外观和操作方式的一种新方法。通过 skin 文件、CSS 文件和站点上服务器控件使用的图像来实现主题功能。所有这些元素都可以构建一个主题，并存储在解决方案的 App_Themes 目录中。



2.2 页面管理

ASP.NET 页面是指带.aspx 扩展名的文本文件，可以被部署在 IIS 虚拟目录树中。页面由代码和标签(tag)组成，它们在服务器上被动态地编译和执行，为提出请求的客户端浏览器(或设备)生成显示内容。对于 Web 开发人员来说，如果想提高页面的运行效率，首先需要了解 ASP.NET 页面是如何组织运行的。

2.2.1 ASP.NET 页面代码模式

ASP.NET 的页面包含两个部分：一部分是可视化元素，包括标签、服务器控件以及一些静态文本等；另一部分是页面的程序逻辑，包括事件处理句柄和其他程序代码。ASP.NET 提供两种模式来组织页面元素和代码：一种是单一文件模式；另一种是后台代码模式。两种模式功能是一样的，可以在两种模式中使用同样的控件和代码，但要注意使用的方式不同，接下来分别进行介绍。

1. 单一文件模式

在单一文件模式下，页面的标签和代码在同一个.aspx 文件中，程序代码包含在<script runat="server"></script>的服务器程序脚本代码块中间，并且代码中间可以实现对一些方法和属性以及其他代码的定义，只要在类文件中可以使用的都可以在此处进行定义。运行时，单一页面被视为继承 Page 类。

2. 后台代码模式

后台代码模式将可视化元素和程序代码分别放在不同的文件中，如果使用 C#语言，则可视



化页面元素为.aspx 文件，程序代码为.cs 文件。使用的语言不同，程序代码后缀也不同。这种模式也被称为代码分离模式。

ASP.NET 4.0 的后台代码分离模式有很大的改进，简单易用且健壮性强。一个典型的代码分离模式的例子如例 2-1 所示。

【例 2-1】代码分离模式示例。

页面文件 Default.aspx 的内容如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>
<html>
<body>
    <h2>
        后台代码模式示例
    </h2>
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
</body>
</html>
```

后台代码程序文件 Default.aspx.cs 的内容如下：

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
public partial class _Default : System.Web.UI.Page
{
        protected void Page_Load(object sender, EventArgs e)
        {
            Label.Text = Request.ServerVariables["ALL_HTTP"];
        }
}
```



ASP.NET 的代码分离模式把一个网页文件分为一个.aspx 文件和一个对应的.aspx.cs 文件，其中，.aspx 文件顶部的页面设置把两个文件联系在一起，如本例中的 CodeFile="Default.aspx.cs"。在

Web 窗体页 Default.aspx 中，添加了一个 Label 服务器控件，在后台代码程序 Default .aspx.cs 中的 Page_Load 事件中设置了该控件的 Text 属性。程序的运行效果如图 2-2 所示。

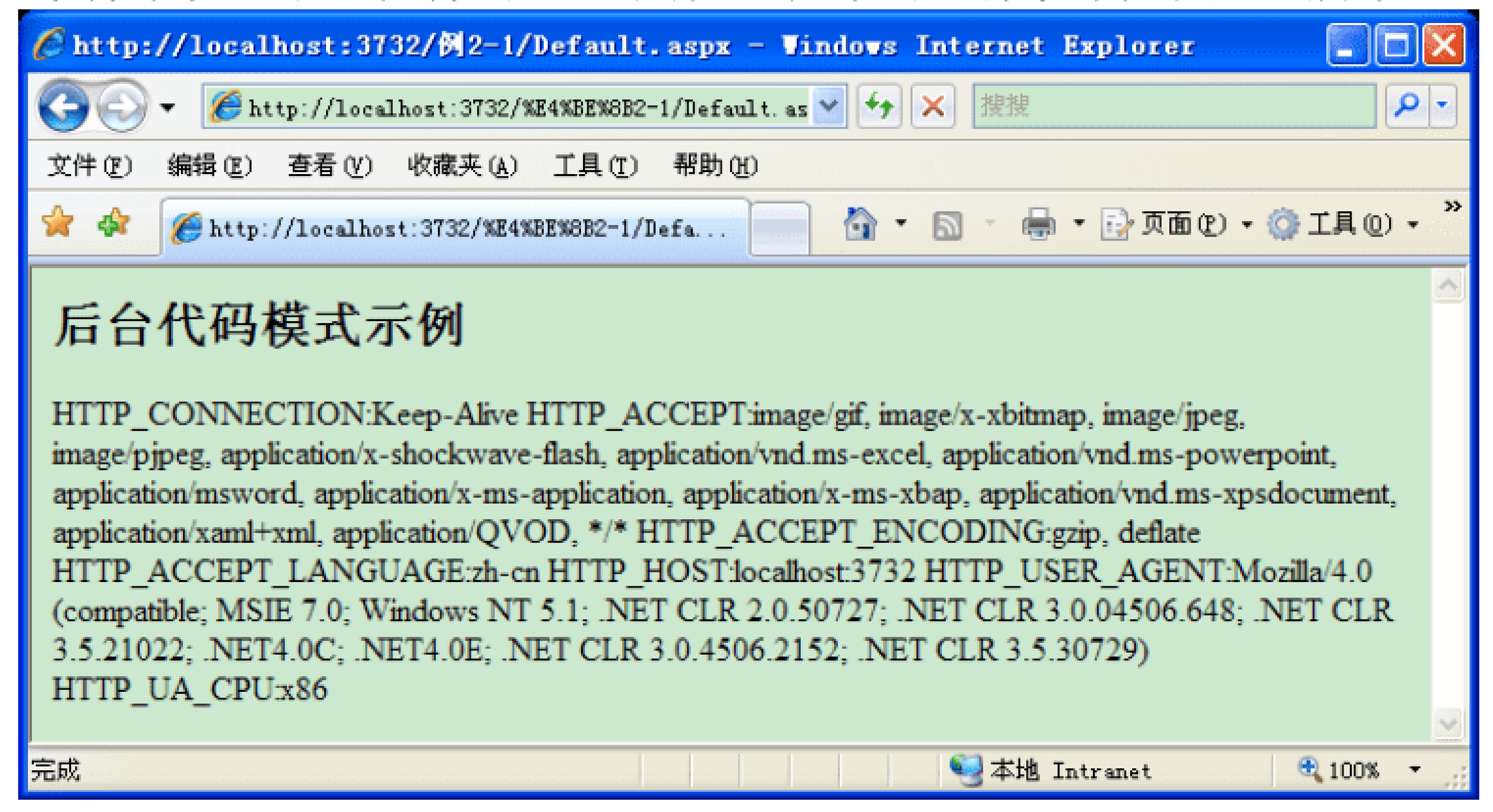


图 2-2　代码分离模式示例

2.2.2　页面往返机制

ASP.NET 网页是以代码形式在服务器上运行的，因此，要使页面中的按钮或其他内容得到处理，必须将该信息提交到服务器。每次页面提交时，都会在服务器端运行其代码，然后把运行的结果呈现给用户。ASP.NET 页面的处理过程循环如下：

(1) 用户通过客户端浏览器请求页面，页面第一次运行，执行初步处理。

(2) 执行的结果以 HTML 标记的形式呈现给浏览器，浏览器对标记进行解释并显示。

(3) 用户输入信息或从可选项中进行选择，或者单击按钮。

(4) 页面将以上用户行为发送到 Web 服务器，在 ASP.NET 中此称为“回发”，也就是页面发送回其自身。例如用户正在访问 Default.aspx 页面，则单击该页面上的某个按钮可以将该页面发送回服务器，发送的目的还是 Default.aspx。

(5) 在 Web 服务器上，该页再次运行，并且使用用户输入或选择的信息。

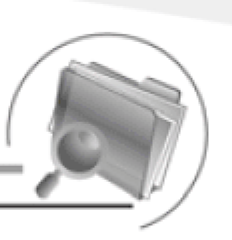
(6) 服务器将执行操作后的页面以 HTML 或 XHTML 标记的形式发送到客户端浏览器。

只要用户访问同一个页面，该循环就会继续。每个循环称为一次“往返行程”。由于页面处理发生在 Web 服务器上，因此，页面可以执行的每个操作都需要一次到服务器的往返行程。

有时可能需要代码仅在首次请求页面时执行，而不是每次回发时都执行，这时可以使用 Page 对象的 IsPostBack 属性来避免对往返行程执行不必要的处理。

2.2.3　页面生存周期

ASP.NET 页面在运行时将经历一个生命周期，在生命周期中，该页面将执行一系列处理步



骤。这些步骤包括初始化、实例化控件、还原和维护状态、运行事件处理程序代码以及进行呈现。

与桌面应用程序中的窗体不同，ASP.NET 网页在使用窗体时不会启动或运行，并且仅当用户单击“关闭”按钮时才会卸载。这是由于 Web 具有断开连接的天性。浏览器从 Web 服务器请求页面时，浏览器和服务器相连的时间仅够处理请求。Web 服务器将页面呈现到浏览器之后，连接即终止。如果浏览器对同一 Web 服务器发出另一个请求，则即使是对同一个页面发出的，该请求仍会作为新请求来处理。

Web 这种断开连接的天性决定了 ASP.NET 页的运行方式。用户请求 ASP.NET 网页时，将创建该页的新实例。如果用户单击按钮以执行回发，将创建该页的一个新实例；该页执行其处理，然后再次被丢弃。这样，每个回发和往返行程都会导致生成该页的一个新实例。



2.3　ASP.NET 的内置对象



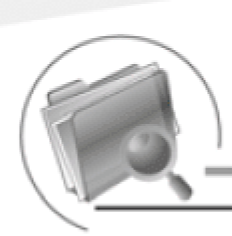
ASP.NET 能够成为一个庞大的软件体系，与它提供了大量的对象类库有很大的关系。这些类库中包含许多封装好的内置对象，开发人员可以直接使用这些对象的方法和属性。这些对象主要包括 Page、Response、Request、Application、Session、Server、ViewState、Cookie 等。下面将分别介绍这些对象的常用属性及方法。

2.3.1　Page 类

在 ASP.NET Framework 中，Page 类为 ASP.NET 应用程序文件所构建的对象提供基本行为。该类在 System.Web.UI 命名空间中，从 TemplateControl 类派生而来，而 TemplateControl 类继承自 System.Web.UI.Control，它也是一种特殊的 Control 类，并实现了 IHttpHandler 接口。

前面介绍了页面往返机制和页面的生存周期。在页面工作过程中，每个页面都被编译为一个类，当有请求的时候就对这个类进行实例化。对于页面的生存周期，Page 对象一共要关心以下 5 个阶段。

- 页面初始化：在这个阶段，页面及其控件被初始化，页面确定这是一个新的请求还是一个回传请求。页面事件处理器 Page_PreInit 和 Page_Init 被调用。另外，所有服务器控件的 PreInit 和 Init 被调用。
- 载入：经过页面初始化之后，页面将进入载入阶段。在该阶段，如果当前页面的请求是一个回传请求，则该页面将从视图状态和控件状态中加载控件的属性。在此过程中，页面将引发 Load 事件。
- 回送事件处理：如果请求是一个回传请求，任何控件的回发事件处理过程都将被调用。
- 呈现：在页面呈现状态中，视图状态被保存到页面。页面和控件的 PreRender 和 Render 方法先后被调用。最后，呈现的结果通过 HTTP 响应发送回客户端。
- 卸载：对页面使用过的资源进行最后的清除处理，控件或页面的 Unload 方法被调用。



在前面介绍页面代码模式时曾说过，单一页面在运行时被视为继承 Page 类。而在后台代码模式中，后台的.cs 文件中包含一个继承自 Page 类的分部类，即具有 partial 关键字的类声明，在对代码分离页进行编译时，ASP.NET 基于.aspx 文件生成一个分部类，该类是.cs 文件中定义的分部类的另一部分，两者一起编译成程序集，运行该程序集可以输出程序到浏览器。

下面介绍 Page 类的常见属性和事件，如表 2-2 所示。

表 2-2　Page 类的常用属性和事件

属性或事件	说　　明
Application	为当前 Web 请求获取 HttpApplicationState 对象
IsPostBack	指示该页是否正为响应客户端回发而加载，或者它是否正被首次加载和访问
IsValid	指示页验证是否成功
Request	获取请求的页的 HttpRequest 对象
Response	获取与该 Page 对象关联的 HttpResponse 对象
Server	获取 Server 对象，它是 HttpServerUtility 类的实例
Session	获取 ASP.NET 提供的当前 Session 对象
Validators	获取请求的页上包含的全部验证控件的集合
ViewState	获取状态信息的字典，这些信息使用户可以在同一页的多个请求间保存和还原服务器控件的视图状态
PreInit	在页初始化开始时发生
PreLoad	在页的 Load 事件之前发生
Load	当服务器控件加载到 Page 对象中时发生
Init	当服务器控件初始化时发生，初始化是控件生存期的第一步
PreRender	在加载 Control 对象之后、呈现之前发生
InitComplete	在页初始化完成时发生
LoadComplete	在页生存周期的加载阶段结束时发生
Unload	当服务器控件从内存中卸载时发生

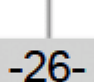
Page 对象的事件贯穿页面执行的整个过程。大多数情况下，只需关心 Page_Load 事件即可。由于 Page_Load 方法在每次页面被加载时执行，所以，即使是回传的情况下也会调用该方法，此时可以使用 Page 对象的 IsPostBack 属性来判断是否回传请求，从而进行不同的处理。

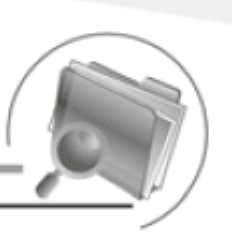
例 2-2 演示了 Page 类各事件发生的时刻。

【例 2-2】演示加载页面时，Page 类各事件的发生顺序。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-2】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动一个 Button 控件和一个 Label





控件到 Web 窗体中，系统自动将其分别命名为 Button1 和 Label1，如图 2-3 所示。

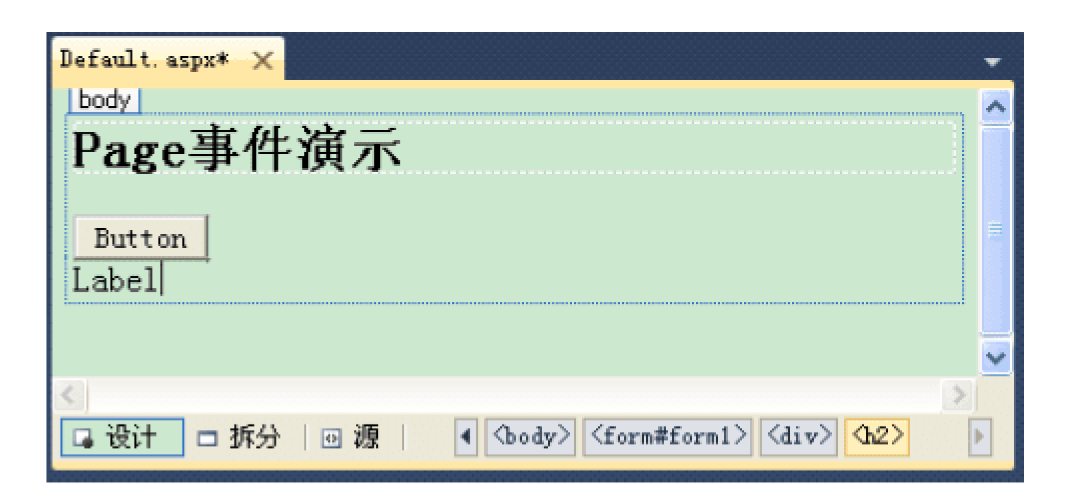


图 2-3　添加控件到 Web 窗体

(3) 双击 Button 控件，为按钮添加单击事件处理程序，同时将跳转到代码文件 Default.aspx.cs 的相应位置，在此添加如下代码，修改 Label 控件的 Text 属性。

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text += "<br>按钮的单击事件处理程序";
}
```

(4) 在代码中同时添加其他的 Page 对象的事件处理程序，如下所示：

```
protected void Page_InitComplete(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 InitComplete 事件";
}
protected void Page_PreInit(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 PreInit 事件";
}
protected void Page_Init(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 Init 事件";
}
protected void Page_PreLoad(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 PreLoad 事件";
}
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text += "<br>Page 对象的 Load 事件";
}
protected void Page_PreRender(object sender, EventArgs e)
{
```



```
        Label1.Text += "<br>Page 对象的 PreRender 事件";
    }
    protected void Page_LoadComplete(object sender, EventArgs e)
    {
        Label1.Text += "<br>Page 对象的 LoadComplete 事件";
    }
```

(5) 单击工具栏中的【启动调试】按钮▶，或者按 F5 键，编译运行程序，在 IE 中显示的运行效果如图 2-4 所示。

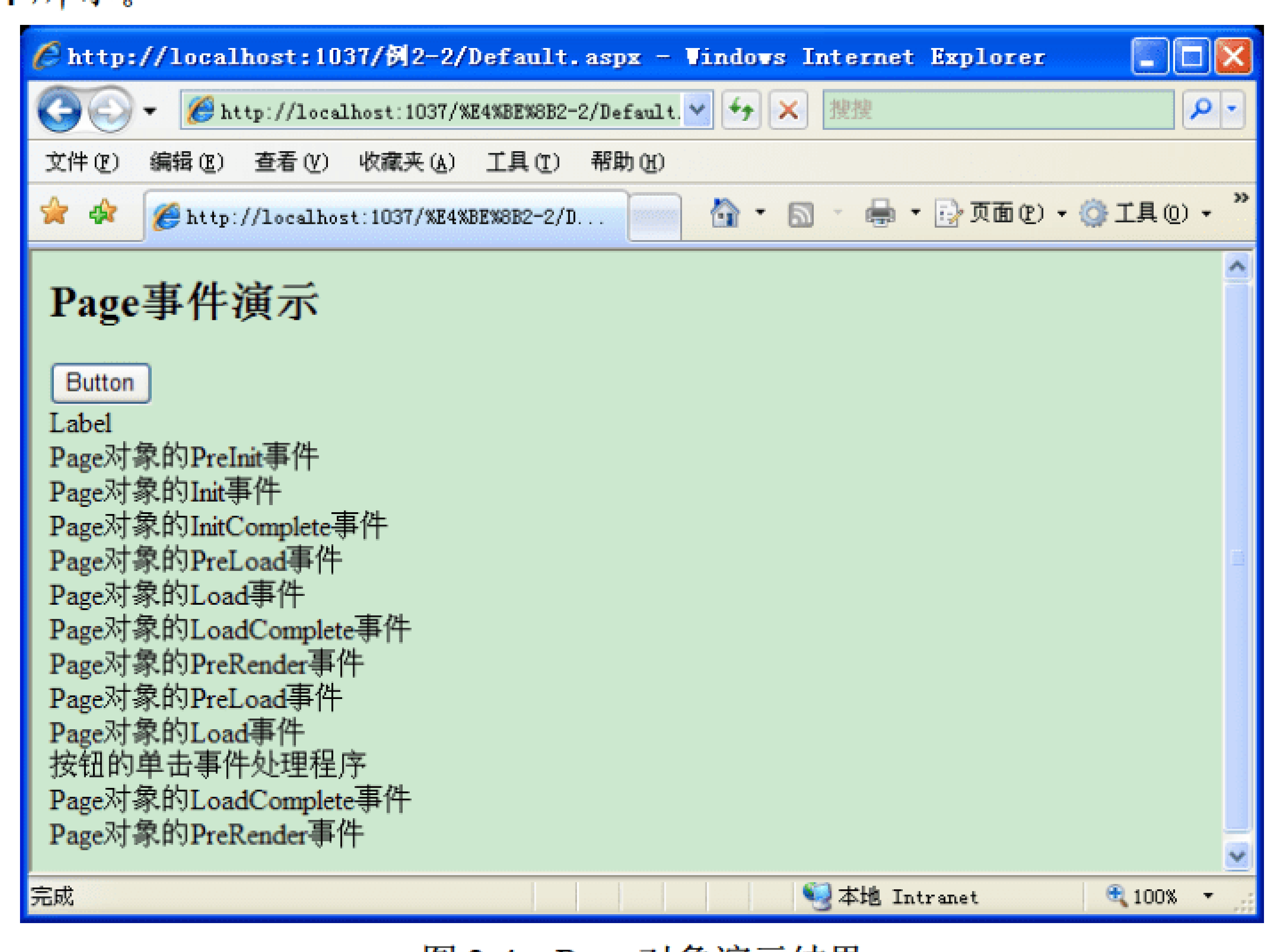


图 2-4　Page 对象演示结果



知识点

Page 对象的事件处理顺序为：PreInit 事件、Init 事件、InitComplete 事件、PreLoad 事件、Load 事件、LoadComplete 事件、PreRender 事件和 Unload 事件。

2.3.2　Request 对象

Request 对象是 ASP.NET 中最有用的对象之一，它与 Response 对象一起使用，达到沟通客户端与服务器端的作用，使它们之间可以很简单地交换数据。

Request 对象接收客户端通过表单或者 URL 地址串发送来的变量，同时，也可以接收其他客户端的环境变量，比如浏览器的基本情况、客户端的 IP 地址等。所有从前端浏览器通过 HTTP 通信协议送往后端 Web 服务器的数据，都是借助 Request 对象完成的。

Request 对象是 System.Web.HttpRequest 类的实例，其常用的属性和方法如表 2-3 所示。

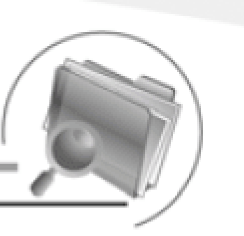


表 2-3　Request 对象的常用属性和方法

属性或方法	说　明
ApplicationPath	获得 ASP.NET 应用程序虚拟目录的根目录
Browser	获取和设置客户端浏览器的兼容性信息
ContentLength	客户端发送信息的字节数
ContentType	获取和设置请求的 MIME 类型
Cookies	获取客户端 Cookie
FilePath	当前请求的虚拟路径
Files	获取客户端上传的文件集合
Form	获取表单变量集合
Headers	获取 HTTP 头信息
HttpMethod	HTTP 数据传输方法，例如 GET、POST
Path	获取当前请求的虚拟路径
PhysicalPath	获取请求的 URL 物理路径
QueryString	获取查询字符串集合
ServerVariables	获取服务器变量集合
TotalBytes	获取输入文件流的总大小
Url	获取当前请求的 URL
UrlReferrer	获取该请求的上一个页面
UserAgent	客户端浏览器信息
UserHostAddress	客户端 IP 地址
UserHostName	客户端 DNS 名称
UserLanguages	客户端语言
BinaryRead	以二进制方式读取指定字节的输入流
MapPath	为当前请求将请求的 URL 中的虚拟路径映射到物理路径
SaveAs	保存 HTTP 请求到硬盘
ValidateInput	验证客户端的输入数据，如果具有潜在的风险，则引发一个异常

ASP.NET 是使用表单(Form)来实现用户数据提交的。对于 HTML 表单，可以使用 Get 方法或 Post 方法来实现数据提交。如果使用 Get 方法，就要使用 Request 对象的 QueryString 集合来获取相关的信息；如果使用 Post 方法，就要使用 Request 对象的 Form 集合来获取相关信息。下面分别讲解如何使用 Get 方法和 Post 方法。

- Get 方法：使用 Get 方法进行数据提时，用户要提交的信息往往是作为查询字符串加在 URL 的后面传给接收程序，一般限制在 2KB 左右。例如：http://www.domain.com/test.aspx?name=myname&password=mypassword。

◉ Post 方法：使用 Post 方法时，用户浏览器的地址栏中不会显示相关的查询字符串。因此，如果需要提交的数据很多时，应使用 Post 方法，因为它对数据的大小和长度没有限制。另外，由于地址栏中不显示相关的查询字符串，所以使用 Post 方法就十分适合用来传递保密信息，例如用户的帐号和密码。

2.3.3 Response 对象

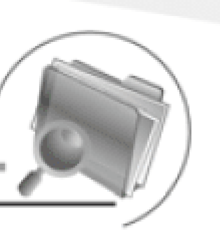
Response 对象实际是在执行 System.Web 命名空间中的 HttpResponse 类。CLR 会根据用户的请求信息建立一个 Response 对象，Response 将用于回应客户端浏览器，告诉浏览器回应内存的报头、服务器端的状态信息以及输出指定的内容。

Response 对象的常用属性和方法如表 2-4 所示。



表 2-4 Response 对象的常用属性和方法

属性或方法	说　明
Buffer	获取或设置是否缓冲输出。如有缓冲，服务器在所有当前处理的页面的语句被处理之前不将 Response 送往客户端，除非有 Flush 或 End 方法被调用。True 表示需要，False 表示不需要，默认值是 True
Cache	获取缓存信息
CharSet	获取和设置输出流的 HTTP 字符集
ContentType	获取和设置输出流的 MIME 类型，默认值为 text/html
Cookie	获取 Cookie 集合
Expires	获取和设置浏览器缓存超时时间
IsClientConnected	获取客户端是否和服务器连接
Status	设置返回给客户端的状态
StatusCode	获取和设置返回给客户端状态字符串
StatusDescription	获取和设置状态说明
AddHeader	添加 HTTP 头信息
AppendCookie	添加一个 Cookie
AppendHeader	添加 HTTP 头信息
AppendToLog	添加自定义信息到 IIS 日志中
BinaryWrite	以二进制的方式输出
Clear	清除输出缓存
Close	关闭和客户端的 Socket 连接
End	发送所有缓冲到客户端，并且停止执行页面
Flush	发送所有缓存到客户端
Redirect	重新定向 URL



(续表)

属性或方法	说　明
SetCookie	更新一个已有的 Cookie
Write	输出信息
WriterFile	直接将指定文件写到输出流

下面的例子演示了 Request 和 Response 对象的配合使用。

【例 2-3】演示利用 Request 对象和 Response 对象进行数据传送。

(1) 在【例 2-2】的网站中添加一个名为 Request.aspx 和一个名为 Response.aspx 的网页。在【解决方案资源管理器】窗口中，右击【例 2-2】解决方案，从弹出的快捷菜单中选择【添加新项】命令，如图 2-5 所示。



图 2-5　选择【添加新项】命令

(2) 在打开的【添加新项】对话框中选择【Web 窗体】，在【名称】文本框中输入文件名，然后单击【添加】按钮，如图 2-6 所示。

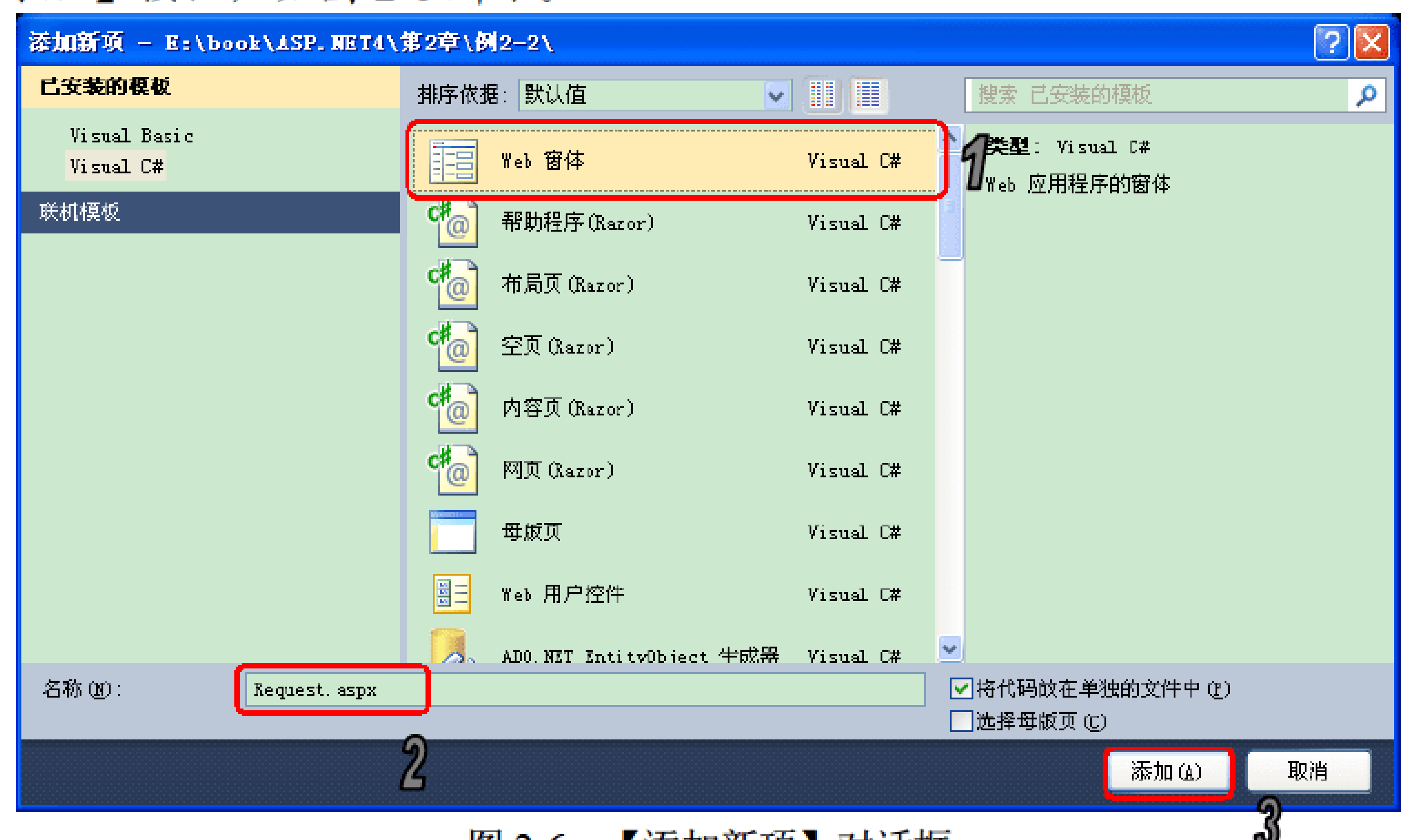


图 2-6　【添加新项】对话框

(3) 在 Request.aspx 页面的<body>标签中添加如下代码:



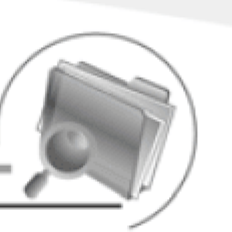
```
<body>
    <h2>登录页面</h2>
    <p><a href="Response.aspx?user=gemm&password=zyd">会员登录</a></p>
    <form id="form1" action="Response.aspx" method="post">
    <p>用户名：<input name="user" type="text" /></p>
    <p>密　码：<input name="password" type="text" /></p>
    <p><input name="submit" type="submit" value="提交" /></p>
    </form>
</body>
```

(4) 在 Response.aspx 页面中的<body>标签中添加如下代码:

```
<body>
    <form id="form1" runat="server">
    <div>
    <asp:button runat="server" text="返回登录页面" onclick="Button1_Click" />
    </div>
    </form>
</body>
```

(5) 在 Response.aspx.cs 代码文件中，添加页面的 Load 事件和按钮的单击事件处理程序，代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.HttpMethod.Equals("GET"))
    {
        Response.Write("以下信息来自于 Request 页面,数据传输方法为 GET<br>");
        Response.Write("QueryString：" + Request.QueryString);
        Response.Write("<br>用户名:" + Request.QueryString["user"]);
        Response.Write("<br>密码:" + Request.QueryString["password"]);
    }
    if (Request.HttpMethod.Equals("POST"))
    {
        Response.Write("以下信息来自于 Request 页面,数据传输方法为 POST<br>");
        Response.Write("<br>用户名:" + Request.Form["user"]);
        Response.Write("<br>密码:" + Request.Form["password"]);
    }
}
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Redirect("Request.aspx");
}
```



(6) 编译并运行程序，在浏览器中加载 Request.aspx 页面，如图 2-7 所示。

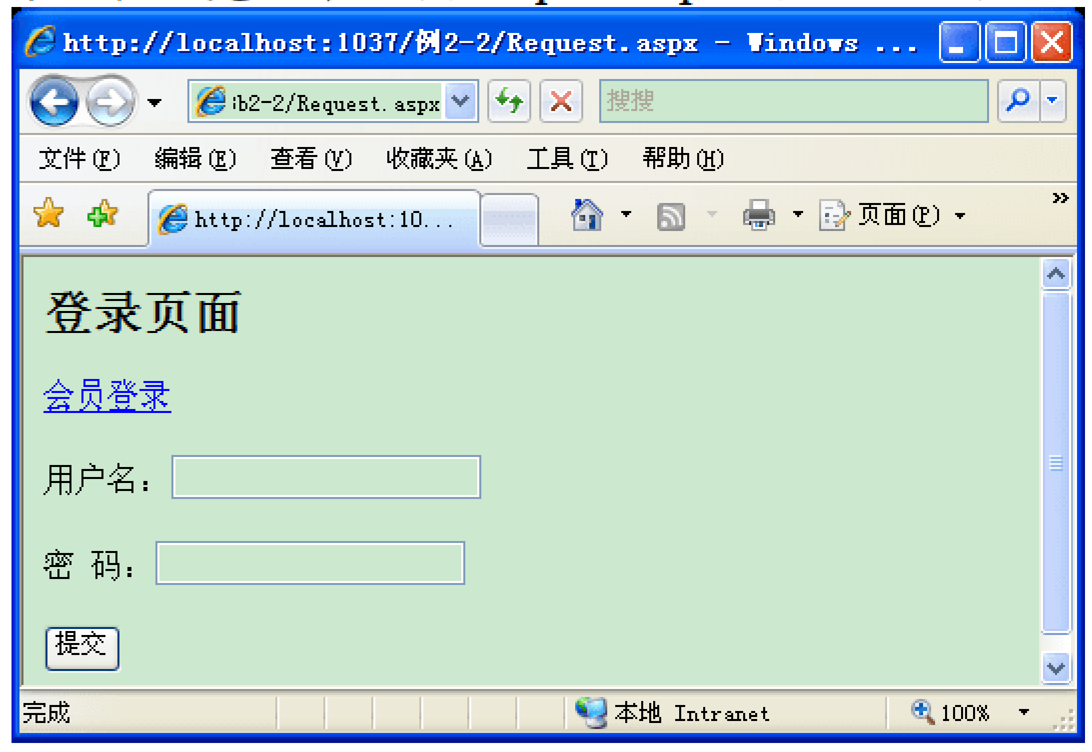


图 2-7　Request.aspx 页面

(7) 单击【会员登录】超链接，跳转到 Response.aspx 页面，可以在【地址栏】中看到传递的参数，如图 2-8 所示。

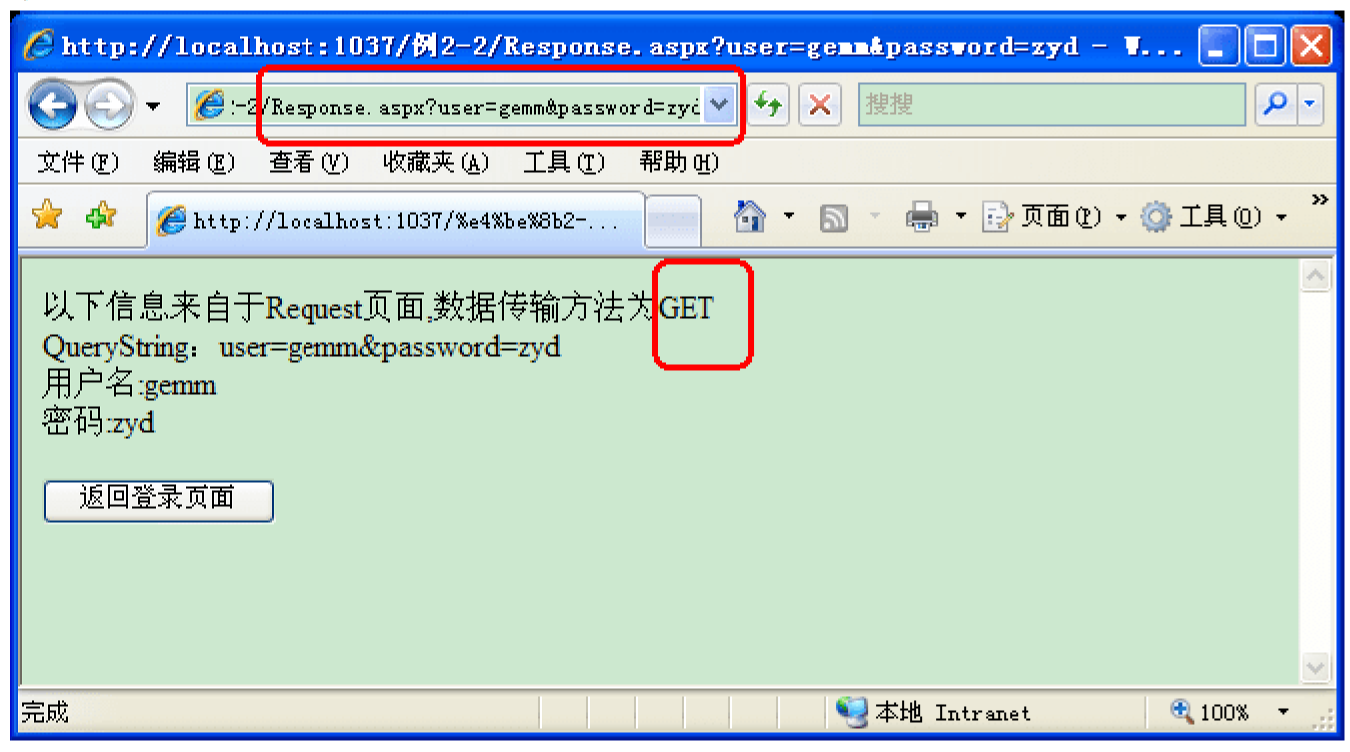


图 2-8　以 GET 方式传送数据

(8) 单击【返回登录页面】按钮，返回到 Request.aspx 页面，在下面的文本框中输入用户名、密码，单击【提交】按钮，此时呈现的页面如图 2-9 所示。

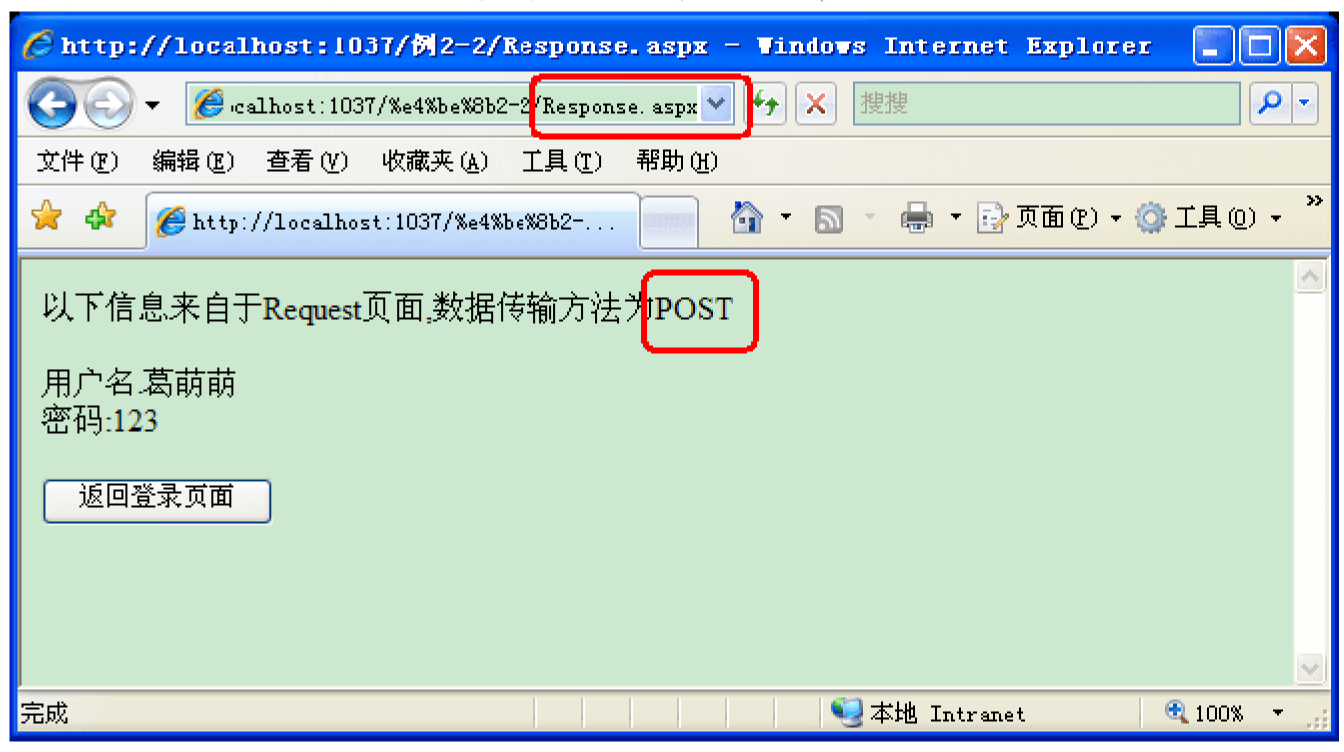


图 2-9　以 POST 方式传送数据



计算机基础与实训教材系列

2.3.4 Application 对象

Application 对象用来保存希望在多个页面之间传递的变量。由于在整个应用程序生存周期中，Application 对象都是有效的，所以在不同的页面中都可以对它进行存取，就像使用全局变量一样方便。

在 ASP.NET 环境中，Application 对象是 System.Web.HttpApplicationState 类的实例，它可以在多个请求、连接之间共享公用信息，也可以在各个请求连接之间充当信息传递的管道。

Application 对象可以建立 Application 变量，它和一般程序变量不同，Application 变量是一个 Contents 集合对象，此变量可以为访问网站的每位用户提供一个共享数据的通道，因为 Application 变量允许网站的每位用户获取或更改其值。

知识点

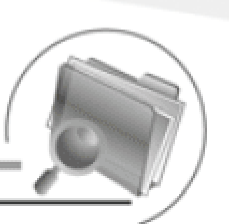
Application 对象是在第一个 Session 对象建立后创建，直到 Web 服务器关机或所有用户都离线后才会删除。

Application 对象的常用属性和方法如表 2-5 所示。



表 2-5　Application 对象的常用属性和方法

属性或方法	说　明
AllKeys	获得访问 HttpApplicationState 集合的所有键
Contents	获得 HttpApplicationState 对象的引用
Count	获得 HttpApplicationState 集合的数量
Item	通过名称和索引访问 HttpApplicationState 集合
Keys	获得访问 HttpApplicationState 集合的所有键，从 NameObjectCollectionBase 继承
StaticObjects	获得所有使用<object>标签声明的应用程序集对象
Add	添加一个新的对象到 HttpApplicationState 集合
Clear	清除 HttpApplicationState 集合中的所有对象
Get	通过索引和名字获得 HttpApplicationState 对象
GetKey	通过索引获得一个 HttpApplicationState 名称
Lock	锁定访问 HttpApplicationState 变量
UnLock	取消锁定，一般情况下需要操作 Application 变量则设置为 Lock，操作完成后则设置为 Unlock
Remove	从 HttpApplicationState 集合删除一个对象
RemoveAll	删除 HttpApplicationState 集合所有对象
RemoveAt	根据索引删除一个 HttpApplicationState 对象
Set	更新一个 HttpApplicationState 变量



下面的例 2-4 演示了 Application 对象的使用。

【例 2-4】演示利用 Application 对象统计网站访问人数。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-4】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动一个 Label 控件到 Web 窗体中，系统自动将它命名为 Label1。

(3) 在 Default.aspx.cs 文件中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Application.Lock();
    Application["usercount"] = (Convert.ToInt32(Application["usercount"]) + 1).ToString();
    Application.UnLock();
    Label1.Text = "您是第" + Application["usercount"].ToString() + "位访客";
}
```



(4) 编译并运行程序，结果如图 2-10 所示，刷新页面可以发现数字会增加。

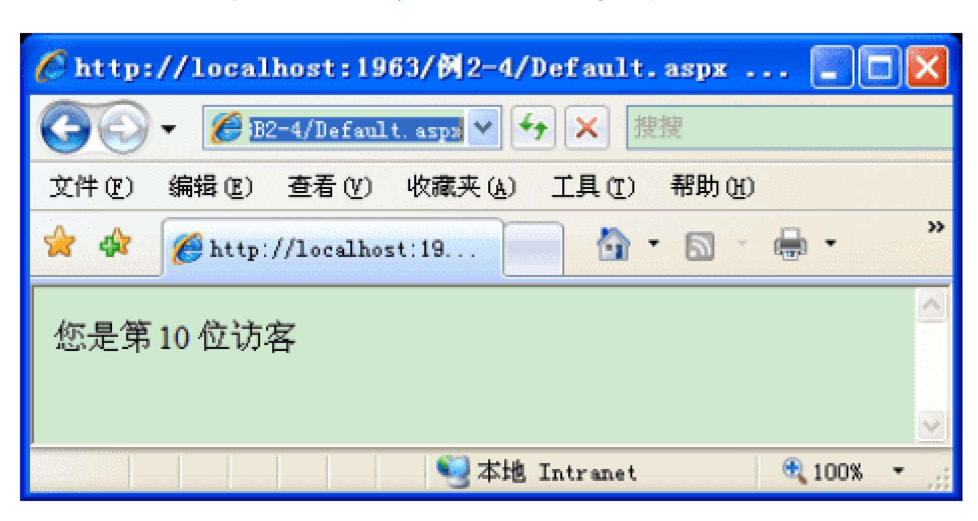


图 2-10　Application 对象示例



提示

语句 Application.Add("key","value")表示向 Application 的 State 集合中加入一个名为 key 的值为 value 的字符串，其效果和 Application("key")="value"以及 Application.Item("key")="value"相同。

2.3.5　Server 对象

Server 对象即服务器对象，就是在服务器上工作的一个对象，它包含一些与服务器相关的信息，是 System.Web.HttpServerUtility 类的实例。使用它可以获取有关最新的出错信息，在页面之间传递控件，对 HTML 进行编码和解码等。

Server 对象提供了许多访问的方法和属性帮助程序有序地执行，其主要属性和方法如表 2-6 所示。

表 2-6　Server 对象的常用属性和方法

属性或方法	说　明
MachineName	获得服务器计算机名称
ScriptTimeout	获得和设置请求超时的事件
ClearError	清除前一个异常
CreateObject	建立一个 COM 组件对象的实例
Execute	执行指定资源并返回
GetLastError	返回前一个异常
Transfer	结束当前页执行，转到其他页执行
HtmlEncode	进行 HTML 编码
HtmlDecode	进行 HTML 解码
MapPath	对虚拟目录进行物理映射
UrlEncode	进行 URL 编码，以便通过 URL 从 Web 服务器到客户端进行可靠的 HTTP 传输
UrlDecode	进行 URL 解码
UrlPathEncode	对 URL 字符串的路径部分进行 URL 编码，并返回已编码的字符串

Server 对象的一个重要功能是对字符进行 URL 和 HTML 的编码和解码。URL 编码的目的是保证所有浏览器能够正确地传输 URL 路径，一些特殊字符如？、&、/、空格和中文字符等，在传输时都有可能使浏览器发生错误。所以有必要先通过编码再将其传输，在需要使用时又通过解码将其还原。HTML 编码的作用是将所有字符全部转换为 HTML 中能够用来显示的字符，例如，<p>如果直接显示就是一个段落，而转换以后就会变成<p>浏览时就可以正确显示出<p>，而不会发生错误。

下面的例 2-5 演示了 Server 对象的使用。

【例 2-5】演示利用 Server 对象获取服务器信息。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-5】，单击【确定】按钮。

(2) 在 Default.aspx.cs 文件中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("下面是利用 Server 对象获取的服务器信息：");
    Response.Write("<br>服务器名称：" + Server.MachineName);
    Response.Write("<br>服务器超时时间：" + Server.ScriptTimeout);
    Response.Write("<br>下面将显示一条水平线<hr>");
    Response.Write("<br>下面将显示字符"+ Server.HtmlEncode(" <hr>"));
}
```

(3) 编译并运行程序，结果如图 2-11 所示。



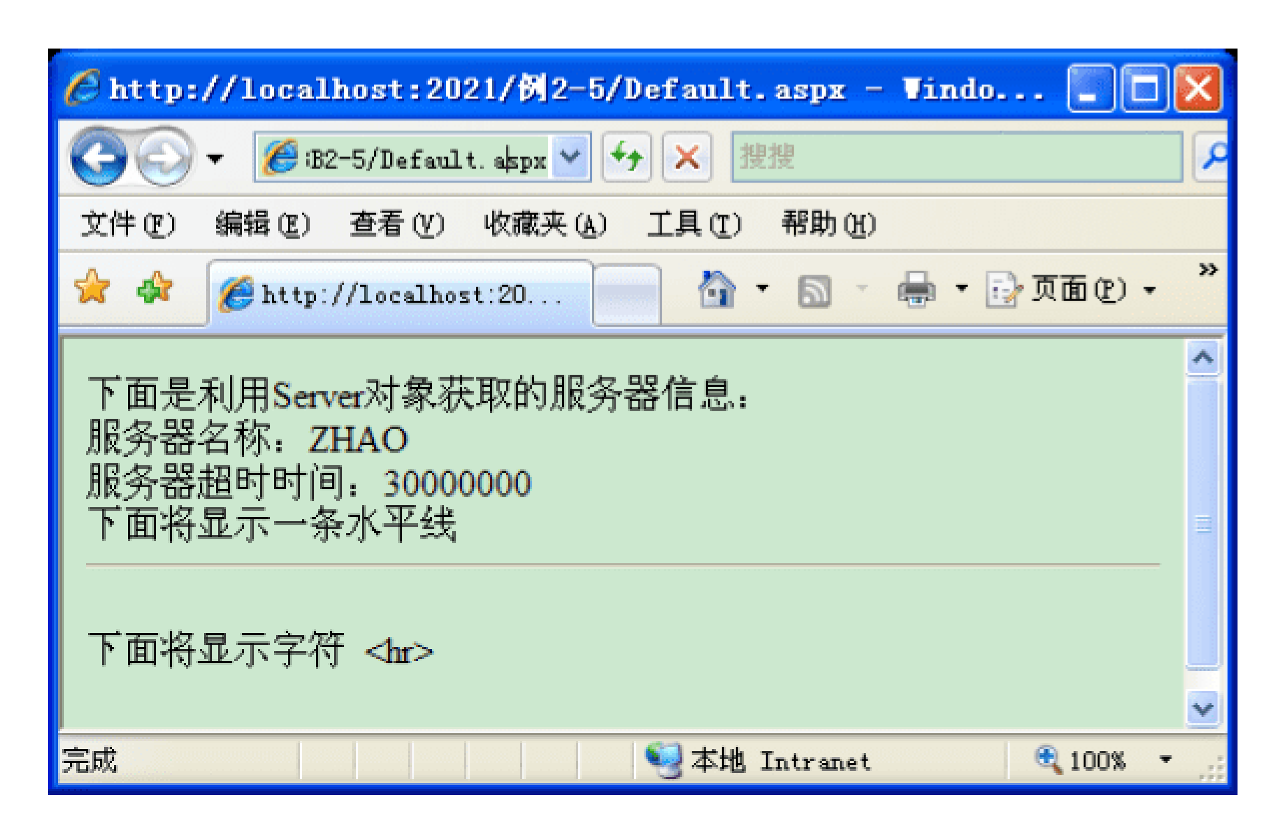


图 2-11　Server 对象示例

2.3.6　Session 对象



Session 对象是 System.Web.HttpSessionState 类的实例，用于储存特定的信息，但是它和 Application 对象在储存信息所使用的对象是完全不同的。Application 对象储存的是共享信息，而 Session 储存的信息是局部的，是随用户不同而不同的。如果只需要在不同页中共享数据，而不是需要在不同的客户端之间共享数据，就可以使用 Session 对象。

Session 的生命周期是有限的(默认值为 20 分钟)，可以使用 Timeout 属性进行设置。在 Session 的生命周期内，Session 的值是有效的。如果用户在大于生命周期的时间里没有再访问应用程序，Session 就会自动过期，Session 对象将会被 CLR 释放，其中保存的数据信息也将丢失。

Session 对象的常用属性和方法如表 2-7 所示。

表 2-7　Session 对象的常用属性和方法

属性或方法	说　明
CodePage	获得或设置字符集标识
Contents	获得当前 Session 状态对象的引用
CookieMode	获得当前的 Cookie 模式，以确定系统是否要将 Session 配置为不需要 Cookie 支持
Count	Session 状态集合的总数
IsCookieless	是否需要 Cookie 支持，如果需要就可以将 Session ID 保存在 Cookie 中，如果不需要就必须嵌入在 URL 中
IsNewSession	标志当前 Session 是否新的 Session
IsReadOnly	是否只读
IsSynchronized	是否同步
Item	通过索引获得或者设置单个 Session 值

(续表)

属性或方法	说　明
Keys	获得 Session 集合的所有键
LCID	获得和设置当前 Session 的本地标识符
Mode	获得当前的 Session 模式
SessionID	获得 Session 的唯一编号，为了区别不同的会话，系统会为每一个会话分配一个唯一的 ID
StaticObjects	获得在 Global.asax 中以<object Runat="Server" Scope="Session" />声明的对象集合
Timeout	获得和设置会话超时时间，如果客户端在连续一个时间段内没有反应，就自动清除会话，断开连接，Timeout 就是这个时间段
Add	添加一个新对象到 HttpApplicationState 集合
Clear	清除 HttpApplicationState 集合中的所有对象
Get	通过索引和名字获得 HttpApplicationState 对象
Abandon	清除当前会话
Copyto	复制 Session 状态集合到一个一维数组
Remove	从 HttpApplicationState 集合删除一个对象
RemoveAll	删除 HttpApplicationState 集合中的所有对象
RemoveAt	根据索引删除一个 HttpApplicationState 对象

下面的例 2-6 演示了 Server 对象的使用。

【例 2-6】演示利用 Server 对象获取服务器信息。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-6】，单击【确定】按钮。

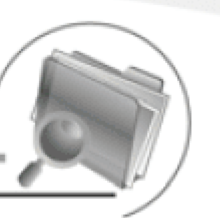
(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动一个 TextBox 控件和一个 Button 控件到 Web 窗体中，系统自动将其分别命名为 TextBox1 和 Button1。

(3) 在 Default.aspx.cs 文件中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("下面是获取到的 Session 信息：");
    Response.Write("<br>SessionID：" + Session.SessionID);
    Response.Write("<br>Session 的数量：" + Session.Count);
    Response.Write("<br>Session 的模式：" + Session.Mode);
    Response.Write("<br>Session 的有效期" + Session.Timeout);
}
```

(4) 为按钮控件添加单击事件处理程序，代码如下所示：

```
protected void Button1_Click(object sender, EventArgs e)
{
```



```
        Session["username"] = TextBox1.Text;
        Response.Redirect("Session.aspx");
    }
```

(5) 添加一个名为 Session.aspx 的网页，在 Session.aspx.cs 的 Load 事件处理程序中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["username"] != null)
    {
        Response.Write("通过 Session 对象获取到的信息："+Session["username"]);
    }
    else
    {
        Response.Write("请在上一页的文本框中输入用户名");
    }
}
```

(6) 编译并运行程序，结果如图 2-12 所示，在文本框中输入信息，单击 Button 按钮，如图 2-13 所示。

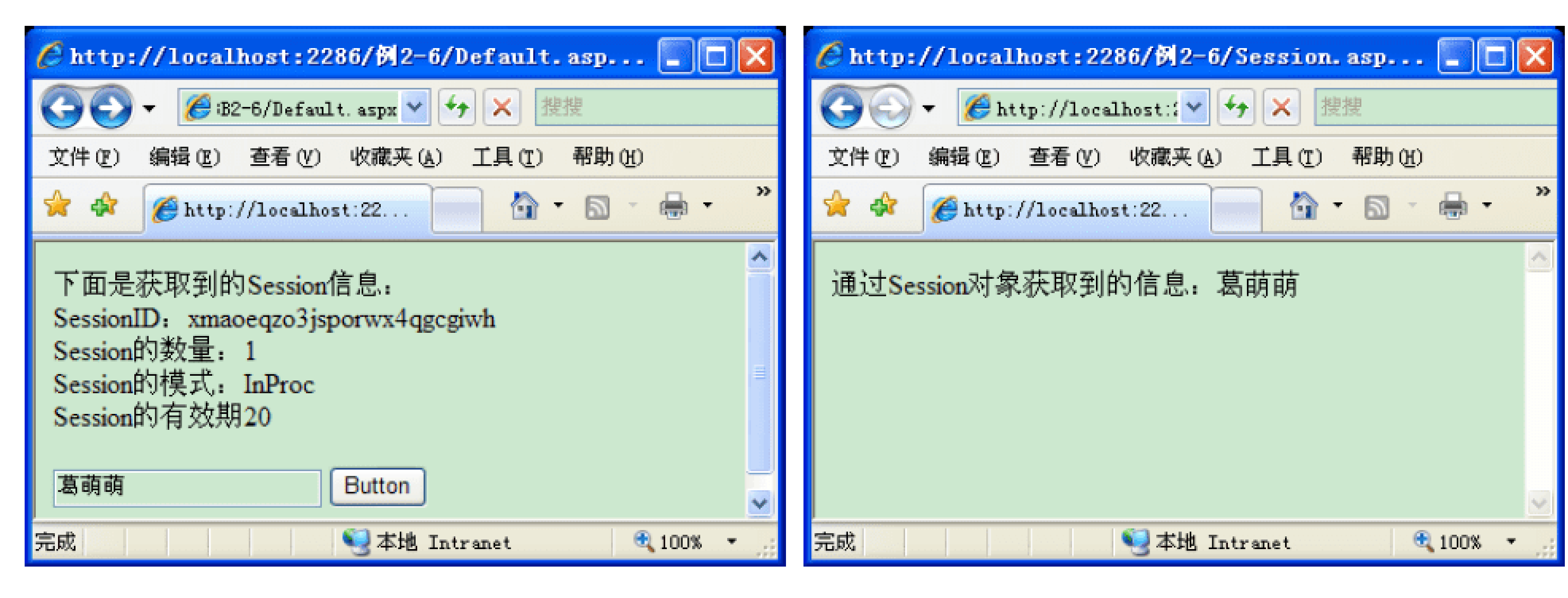


图 2-12　Session 对象示例　　　　图 2-13　通过 Session 对象获取信息

2.3.7　ViewState 对象

ViewState(视图状态)对象是 Page 对象的一个属性，是状态管理中常用的一种对象，可以用来保存页和控件的值。视图状态是 ASP.NET 页框架默认情况下用于保存往返过程之间的页面信息以及控件值的方法。





知识点

视图状态中存储的常见数据类型有字符串、整数、布尔值、Array 对象、ArrayList 对象、哈希表和泛型对象等。

当呈现页的 HTML 形式时，需要在回发过程中保留的页的当前状态和值将被序列化为 Base64 编码的字符串，并输出到视图状态的隐藏字段中。可以通过实现自定义的 PageStatePersiste 类存储页数据，也可以更改默认行为并将视图状态存储到另一个位置，如 SQL Server 数据库。

程序员可以通过使用页面的 ViewState 属性将往返过程中的数据保存到 Web 服务器端，然后利用自己的代码访问视图状态。ViewState 属性是 StateBag 类的实例，它是一个包含键/值对的字典，并通过唯一的键名来访问对应的值。

使用 ViewState 可以带来很多方便，但是也有一些问题是需要注意的。



- 视图状态提供了 ASP.NET 页面的特定状态信息。如果需要在多个页上使用信息，或者需要在访问网站时保留信息，则应当使用另一种方法(如应用程序状态、会话状态或个性化设置)来维护状态。
- 视图状态信息将序列化为 XML，然后进行 Base64 编码，这将生成大量的数据。将页回发到服务器时，如果视图状态包含大量信息，则会影响页的性能。
- 虽然使用视图状态可以保存页和控件的值，但是在某些情况下，需要关闭视图状态。如使用 GridView 控件显示数据，单击 GridView 控件的下一页按钮，此时，GridView 控件呈现的数据已经不再是前一页的数据，那么如果使用视图状态将前一页数据保存下来，不仅没有必要而且还会生成大量隐藏字段，增大页面大小。

如果隐藏字段中的数据量过大，某些代理的防火墙将禁止访问包含这些数据的页。由于所允许的最大数据量随所采用的防火墙和代理的不同而不同，因此大量隐藏字段可能会导致偶发性问题。为了避免这一问题，可采取以下措施：如果 ViewState 属性中存储的数据量超过了页的 MaxPageStateFieldLength 属性中指定的值，该页会将视图状态拆分为多个隐藏字段，可以使每个单独字段的大小在防火墙拒绝的大小之下。

下面的例 2-7 演示了 ViewState 的使用。

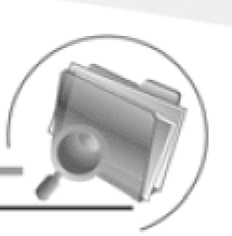
【例 2-7】ViewState 对象的使用。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-7】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动 3 个 Label 控件、2 个 TextBox 控件和 2 个 Button 控件到 Web 窗体中，控件的 Text 属性和页面布局如图 2-14 所示。



图 2-14　页面布局及控件的 Text 属性



(3) 在 Default.aspx.cs 文件中添加页面的 Load 事件和两个按钮的单击事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        ViewState.Add("name", "赵艳铎");
        ViewState.Add("age", 30);
    }
}
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != "")
        ViewState["name"] = TextBox1.Text;
    if (TextBox2.Text != "")
        ViewState["age"] = TextBox2.Text;
}
protected void Button2_Click(object sender, EventArgs e)
{
    Label3.Text = "ViewState 信息如下:<br>姓名：";
    Label3.Text += ViewState["name"];
    Label3.Text += "<br>年龄：";
    Label3.Text += ViewState["age"];
}
```

(4) 编译并运行程序，单击【读取视图状态】按钮，读取 ViewState 的初值，如图 2-15 所示；输入姓名和年龄后，单击【保存视图状态】按钮，然后再次单击【读取视图状态】按钮，读取 ViewState 的新值，如图 2-16 所示。

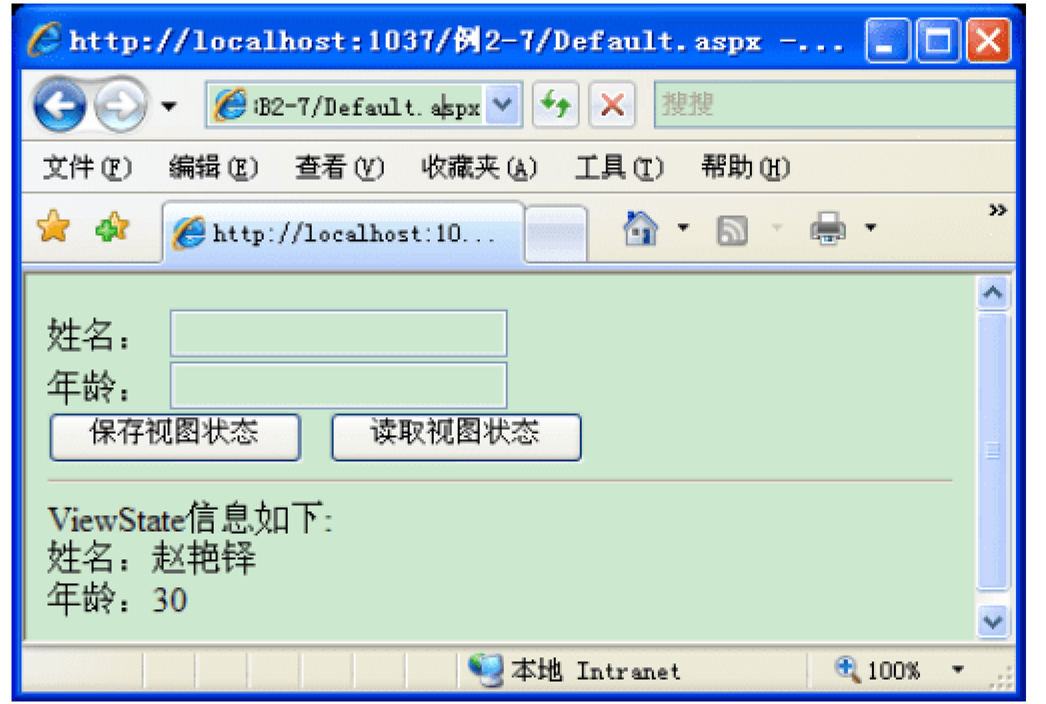


图 2-15　读取视图状态的初值



图 2-16　读取 ViewState 中的新值

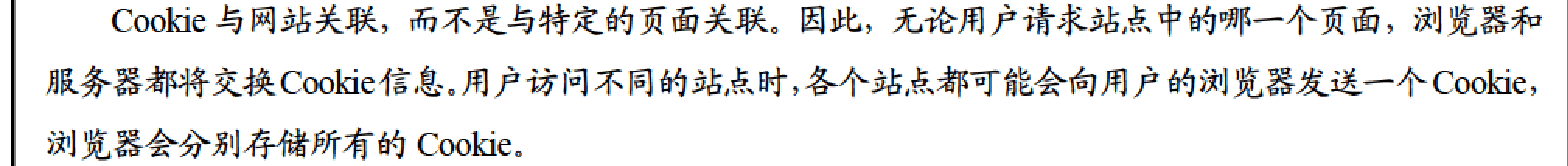
2.3.8 Cookie 对象

Cookie 对象是比较常用的一种对象，通常用来存储少量浏览者的信息，如浏览者的喜好、用户名、Email 地址等信息，以便当浏览者再次登录网站时，不必再次填写这些信息。

1. Cookie 对象简介

Cookie 其实只是一些小文本，将一些用户信息储存在客户端的机器中，它全部存储于 Windows 目录下的 Cookie 文件夹中，以便于在每次请求时被服务器在设定的时间内进行读取。Cookie 的大小是有限制的，一般浏览器会将其控制在 4096 个字节以内。


提示



Cookie 与网站关联，而不是与特定的页面关联。因此，无论用户请求站点中的哪一个页面，浏览器和服务器都将交换Cookie信息。用户访问不同的站点时，各个站点都可能会向用户的浏览器发送一个Cookie，浏览器会分别存储所有的 Cookie。

使用 Cookie 具有如下优点。

- 可配置到期规则：Cookie 可以在浏览器会话结束时到期，或者可以在客户端计算机上无限期存在，这取决于客户端的到期规则。
- 不需要任何服务器资源：Cookie 存储在客户端并在发送后由服务器读取。
- 简单性：Cookie 是一种基于文本的轻量结构，包含简单的键/值对。
- 数据持久性：虽然客户端计算机上 Cookie 的持续时间取决于客户端上的 Cookie 过期处理和用户干预，但 Cookie 通常是客户端上持续时间最长的数据保留形式。

2. Cookie 的设置与获取

浏览器向服务器发出请求时，会随请求一起发送该服务器的 Cookie。ASP.NET 应用程序是通过使用 Request 和 Response 对象的 Cookies 集合对象来读取和设置 Cookie 的。

服务器不能直接修改 Cookie，如果要修改 Cookie，必须先创建一个具有新值的 Cookie。例如：

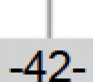
```
HttpCookie cookie=new HttpCookie("name")
```

上述代码创建了一个名为 name 的 HttpCookie 实例。

建立实例后，再为其赋值。在一个 Cookie 中可以存储一个值，也可以储存多个值。通过设置 Cookie 的 Value 属性值，可以在 Cookie 中储存一个值，代码如下：

```
Cookie.Value="葛萌萌"
```

通过 Cookie 的 Values 集合，可以在同一个 Cookie 中储存多个值。例如：





```
HttpCookie cookie=new HttpCookie("student");
Cookie.Values.Add("Admin", "赵艳铎");
Cookie.Values.Add("Member1", "葛萌萌");
Cookie.Values.Add("Member2", "小石头");
```

Values 集合的 Add 方法中第一个参数为关键字(Key)，第二个参数是设置的值(Value)。

由于 Cookie 在用户计算机中，因此无法通过程序将其直接移除。但是，可以让浏览器来删除 Cookie，具体做法是创建一个与要删除的 Cookie 同名的新 Cookie，并将该 Cookie 的到期日期设置为过去的某个日期，当浏览器检查 Cookie 的到期日期时，便会丢弃这个已过期的 Cookie。

3. 确定浏览器是否接受 Cookie

浏览器除了限制 Cookie 的大小，还限制站点可以在用户计算机上存储的 Cookie 的数量。大多数浏览器只允许每个站点存储 20 个 Cookie，如果试图存储更多的 Cookie，则最早存放的 Cookie 便会被覆盖掉。有些浏览器还会对接受的所有 Cookie 总数做出限制，通常为 300 个。



另外，用户还可以将浏览器设置为拒绝接受 Cookie。设置为拒绝接受 Cookie 后，虽然不能向客户端写入 Cookie 信息，但是不会引发任何错误。同样，浏览器也不向服务器发送有关 Cookie 的任何信息。确定 Cookie 是否被接受的一种方法是尝试编写一个 Cookie，然后再尝试读取该 Cookie。如果无法读取已编写的 Cookie，则可以确定浏览器不接受 Cookie。

下面的例 2-8 演示了 Cookie 的使用。

【例 2-8】演示 Cookie 的设置与读取。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-8】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动 1 个 Label 控件、1 个 TextBox 控件和 1 个 Button 控件到 Web 窗体中，控件的 Text 属性和页面布局如图 2-17 所示。

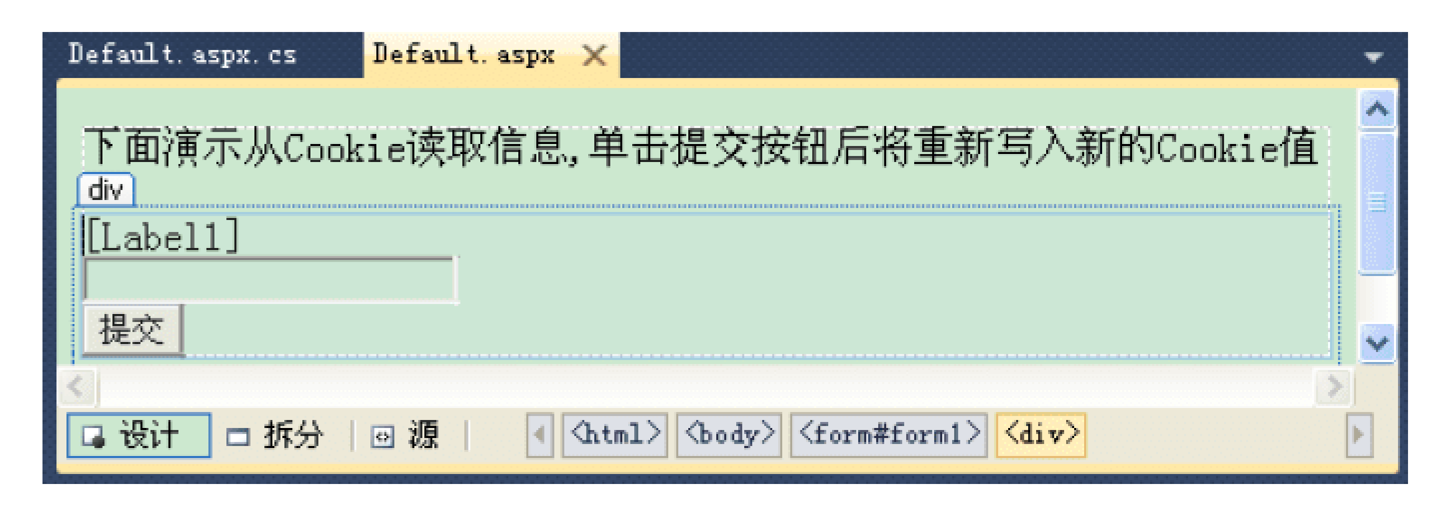


图 2-17　页面布局及控件的 Text 属性

(3) 在 Default.aspx.cs 文件中添加页面的 Load 事件和按钮的单击事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    HttpCookie cookie = Request.Cookies["user"];
    if (cookie == null)
```

```
            Label1.Text = "欢迎进入 ASP.NET 4.0 世界";
        else
            Label1.Text = "欢迎  " + cookie.Value + "  进入 ASP.NET 4.0 世界";
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (TextBox1.Text == "")
        {
            Label1.Text = "请在下面的文本框中输入用户名!";
            return;
        }
        HttpCookie cookie = Request.Cookies["user"];
        if (cookie == null)
           cookie = new HttpCookie("user");
        cookie.Value=TextBox1.Text;
        cookie.Expires = DateTime.Now.AddDays(1);
        Response.Cookies.Add(cookie);
    }
```

(4) 编译并运行程序，初次访问该页面时，由于 Cookie 中没有值，所以显示效果如图 2-18 所示。在文本框中输入一个值，单击【提交】按钮，将该值保存到 Cookie 中。

(5) 按 F5 键刷新页面，可以看到页面发生了变化，刚才保存的 Cookie 值被读取出了，如图 2-19 所示。

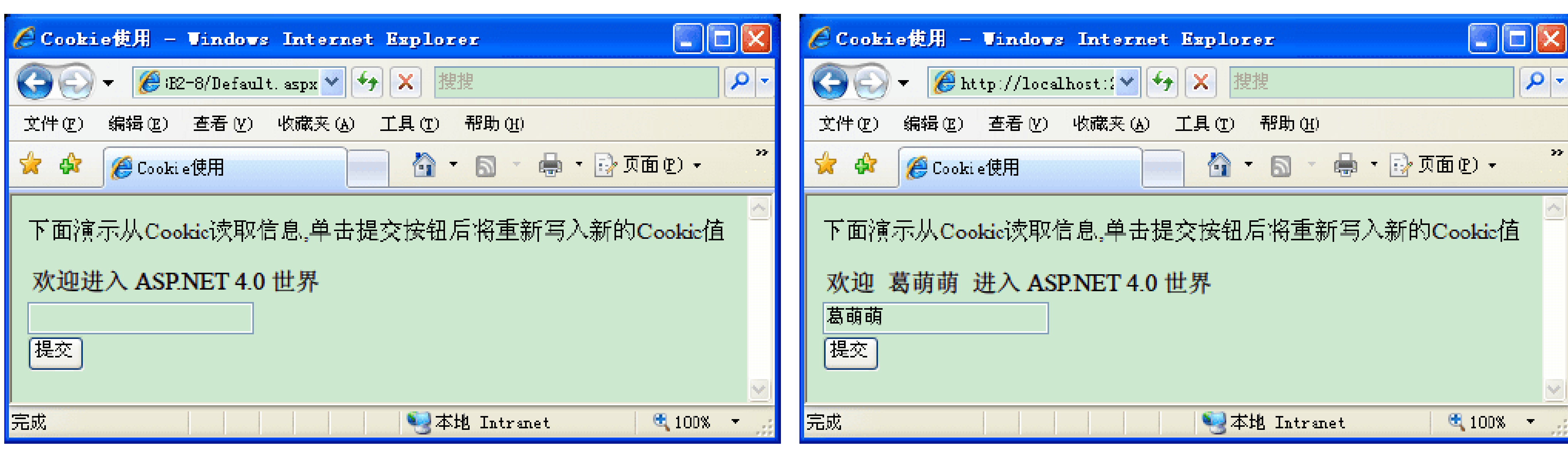
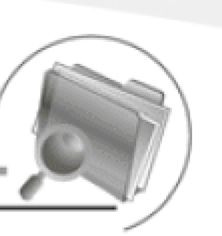


图 2-18　页面初始效果　　　　图 2-19　读取到的 Cookie 值

(6) 接下来看一下 Cookie 到底是怎么保存的。在浏览器窗口中选择【工具】|【Internet 选项】命令，打开【Internet 选项】对话框，如图 2-20 所示。

(7) 单击【浏览历史记录】选项区域中的【设置】按钮，打开【Internet 临时文件和历史记录设置】对话框，如图 2-21 所示。



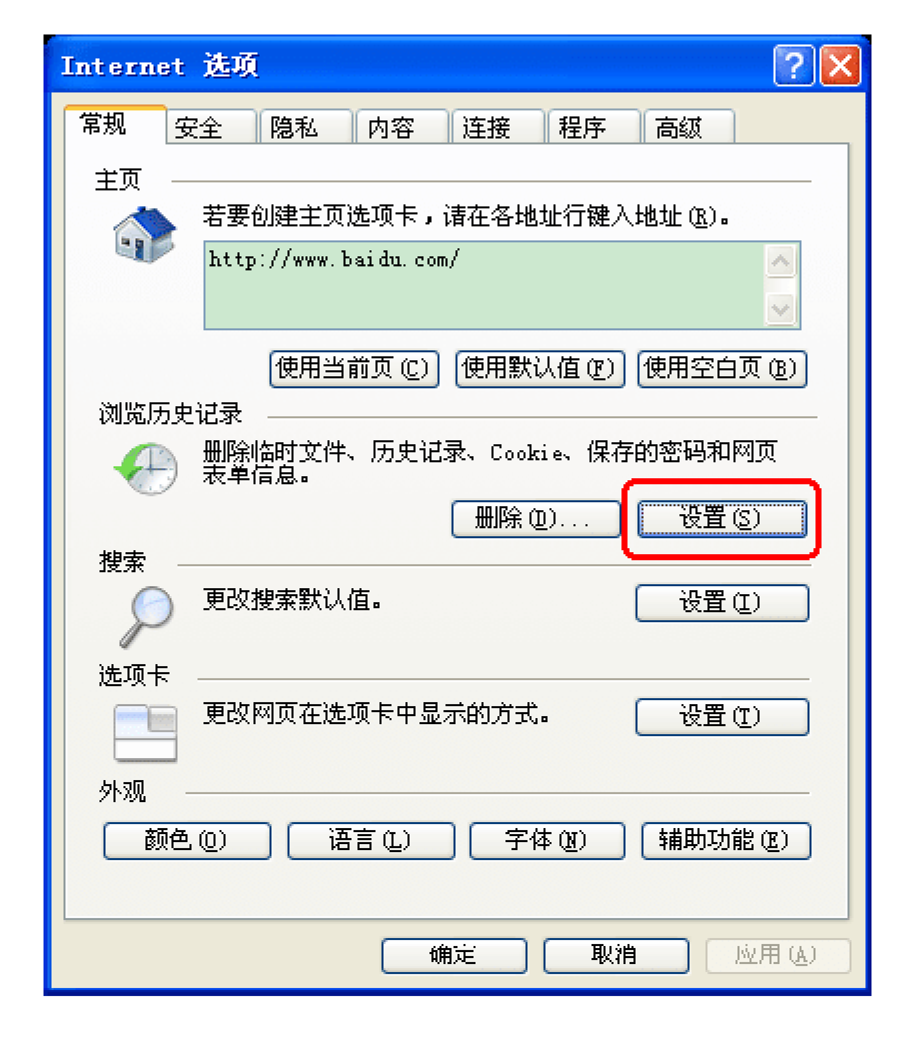


图 2-20　【Internet 选项】对话框

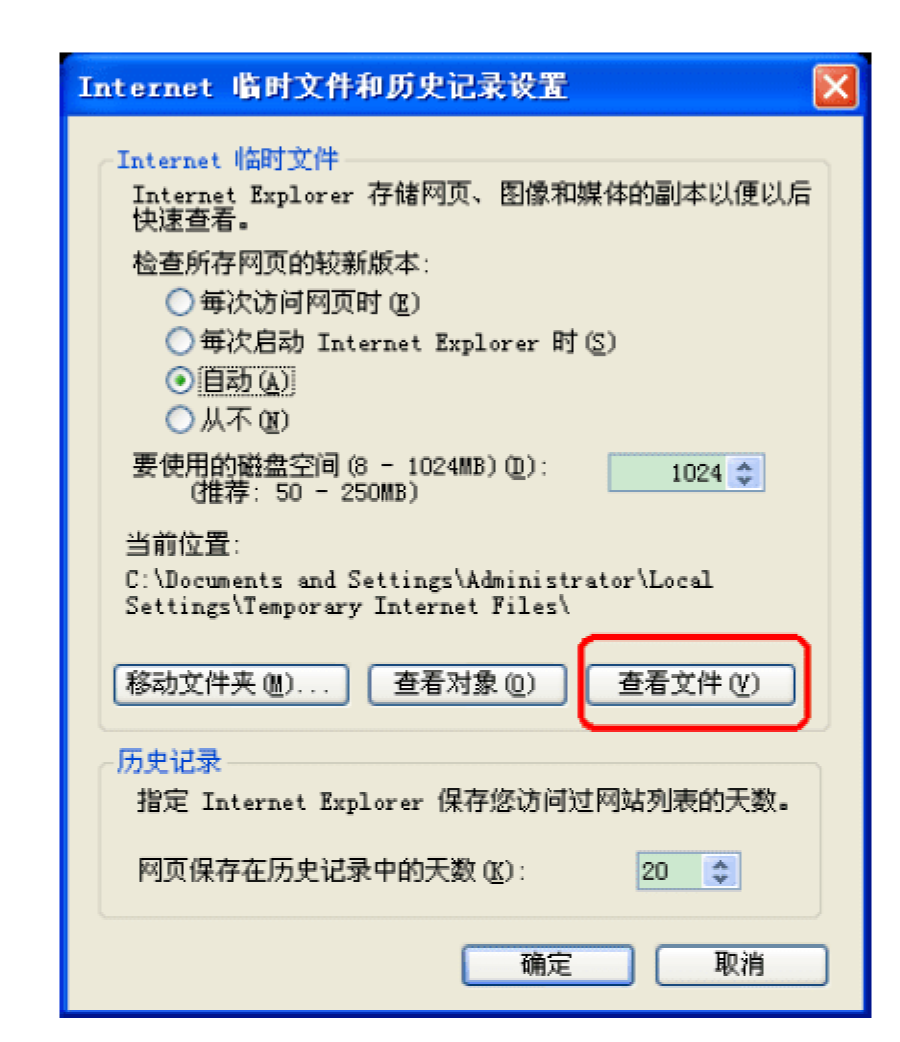


图 2-21　【Internet 临时文件和历史记录设置】对话框



(8) 单击【查看文件】按钮，打开 Windows 资源管理器，定位到 Cookie 存放的目录，如图 2-22 所示，在此目录有很多文本文件和一些图片文件等 Internet 临时文件。

(9) Cookie 文件一般都是以 cookie 打头、以网站域名为结尾的文件，如本例网站的 Cookie 文件在笔者的电脑上是“cookie:administrator@localhost/”。

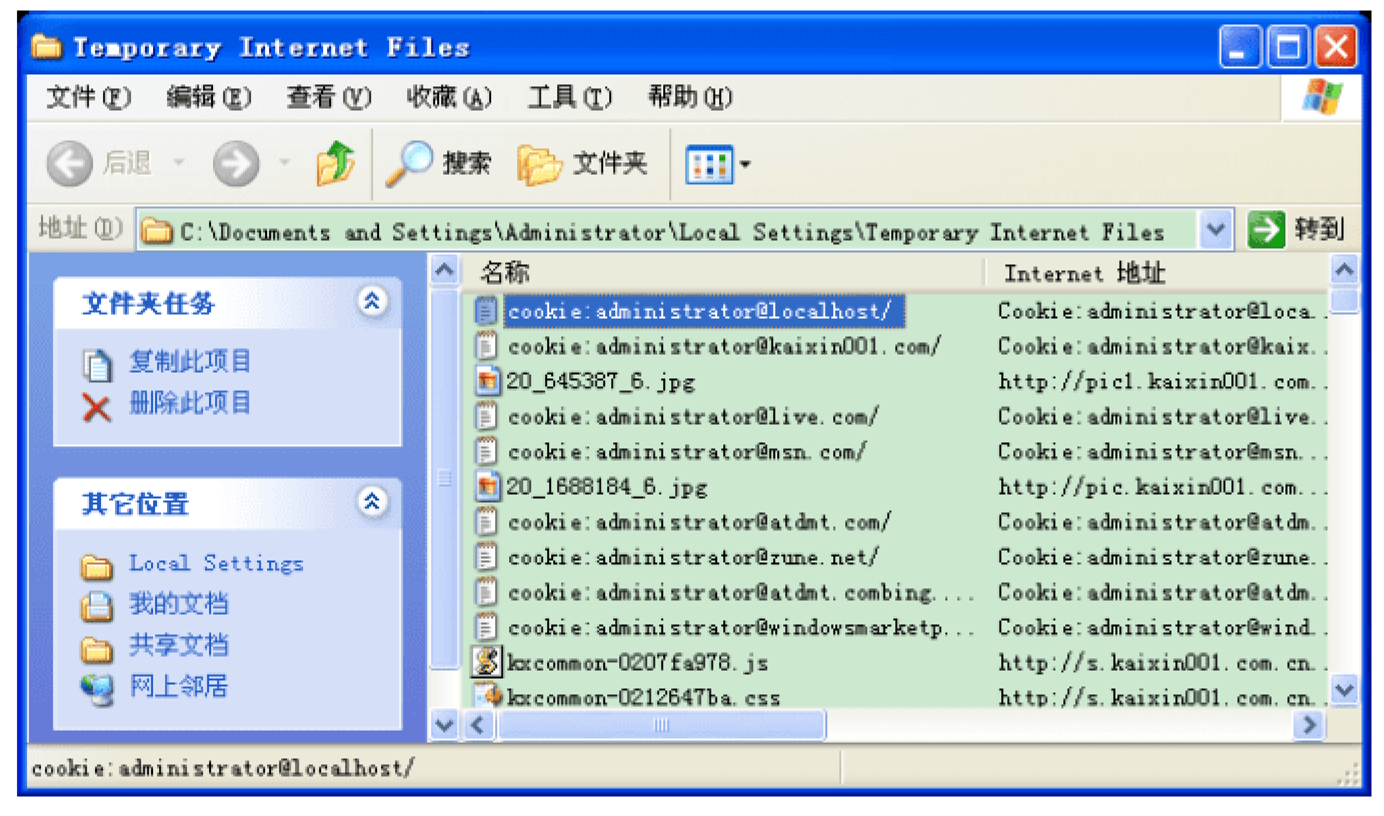


图 2-22　打开 Cookie 存放的目录

(10) 双击 Cookie 文件，可以通过文本编辑器打开并查看该文件，如图 2-23 所示。

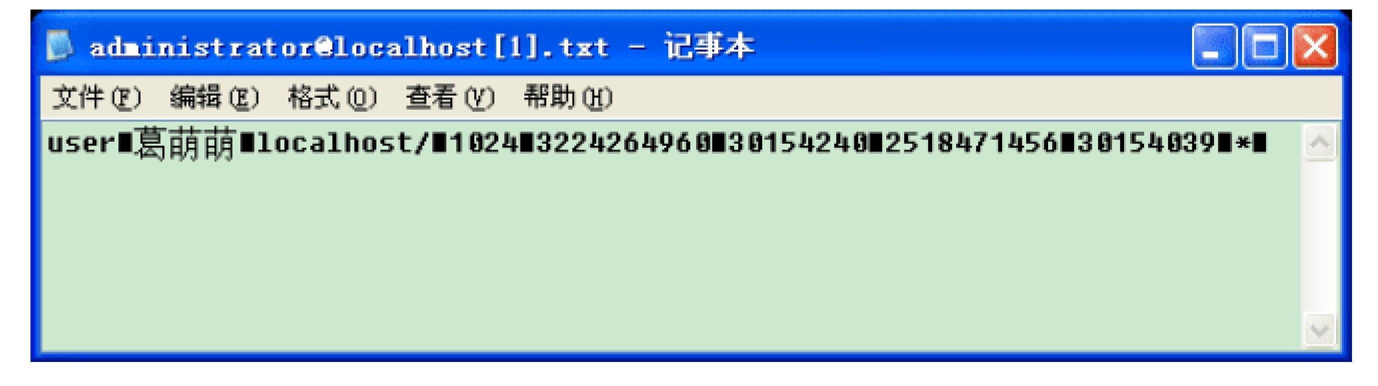
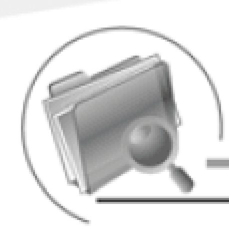


图 2-23　查看 Cookie 文件

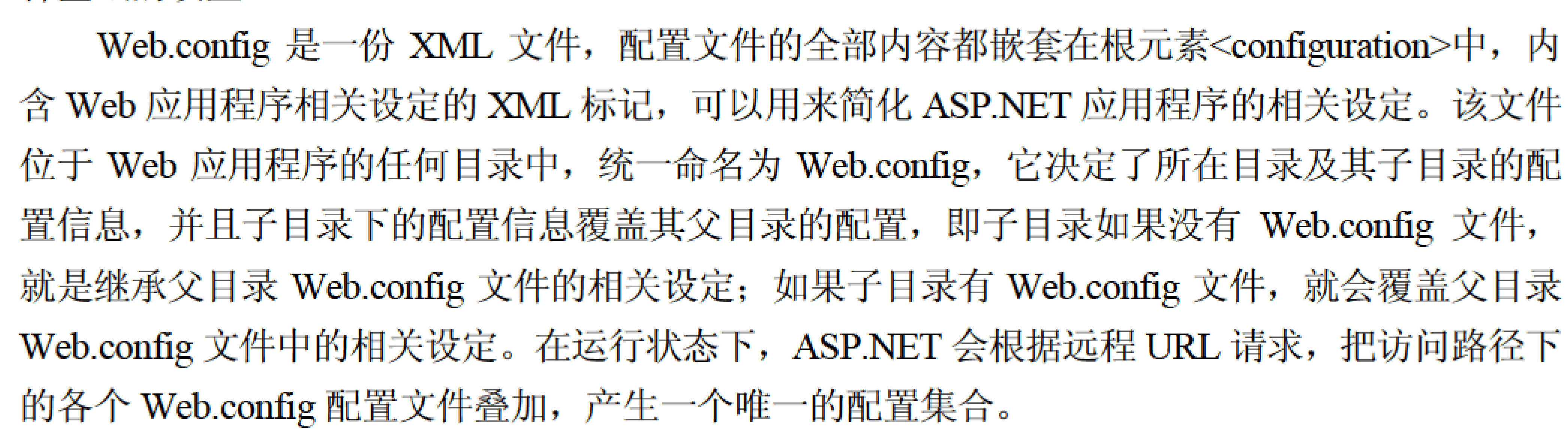


2.4 ASP.NET 配置管理

使用 ASP.NET 配置系统的功能，可以配置整个服务器上的所有 ASP.NET 应用程序、单个 ASP.NET 应用程序和各个页面或应用程序子目录，也可以配置各种具体的功能，如身份验证模式、页缓存、编译器选项、自定义错误、调试和跟踪选项等。

2.4.1 web.config 文件介绍

每一个 Web 应用程序都包含一个 Web.config 配置文件，该配置文件为 ASP.NET 提供了各种基础的设置。



Web.config 是一份 XML 文件，配置文件的全部内容都嵌套在根元素<configuration>中，内含 Web 应用程序相关设定的 XML 标记，可以用来简化 ASP.NET 应用程序的相关设定。该文件位于 Web 应用程序的任何目录中，统一命名为 Web.config，它决定了所在目录及其子目录的配置信息，并且子目录下的配置信息覆盖其父目录的配置，即子目录如果没有 Web.config 文件，就是继承父目录 Web.config 文件的相关设定；如果子目录有 Web.config 文件，就会覆盖父目录 Web.config 文件中的相关设定。在运行状态下，ASP.NET 会根据远程 URL 请求，把访问路径下的各个 Web.config 配置文件叠加，产生一个唯一的配置集合。

举例来说，一个对 URL:http://localhost/website/ownconfig/test.aspx 的访问，ASP.NET 会根据以下顺序来决定最终的配置情况：

(1) .\Microsoft.NET\Framework\{version}\Web.config (默认配置文件)

(2) .\webapp\Web.config(应用的配置)

(3) .\webapp\ownconfig\Web.config(自己的配置)

1. 配置文件的语法规则

Web.config 的全部内容都被置于标记<configuration>和</configuration>之间。在该文件中，XML 标记的属性就是设定值，标记名称和属性值格式是字符串，第一个单词开头字母是小写，之后每一个单词的首字母大写，例如<appSetting>。Web 配置文件的示例如下所示：

```
<configuration>
    <appSettings>
      <add key="dbType" value="Access Database"/>
    </appSettings>
    <connectionsStrings>
      <add name="provider" connectionString="Microsoft.Jet.OLEDB.4.0;"/>
      <add name="database" connectionString="/chart7/Exam.mdb"/>
    </connectionsStrings>
    <system.web>
```



```
            <sessionState cookieless="false" timeout="10"/>
            <compilation defaultLanguage="C#" debug="true"/>
            <globalization fileEncoding="gb2312" requestEncoding="gb2312" culture= "zh-CN"/>
            <customErrors mode="RemoteOnly"/>
        </system.web>
    </configuration>
```

可以看到，这段配置信息是一个基于 XML 格式的文件，根标记是<configuration>，所有的配置信息均被包含在<configuration>及</configuration>标记中，其子标记<appSettings>、<connectionsStrings>和<system.web>是各设定区段。在<system.web>下的设定区段属于 ASP.NET 相关设定。

在 Web 配置文件的<appSettings>区段可以创建 ASP.NET 程序所需要的参数，每个<add>标记可以创建一个参数，属性 key 是参数名称，value 是参数值。ASP.NET 2.0 以后新增了<connectionStrings>区段，可以指定数据库连接字符串，在<connectionStrings>标记的<add>子标记也可以创建连接字符串，属性 name 是名称，connectionStrings 是连接字符串的内容。



2. 常用区段标记

Web.config 文件的常用区段标记如表 2-8 所示。

表 2-8　常用设定区段标记说明

设 定 区 段	说　　明
<AnonymousIdentification>	控制 Web 应用程序的匿名用户
<Authentication>	设定 ASP.NET 的验证方式
<Authorization>	设定 ASP.NET 用户授权
<BrowserCaps>	设定浏览程序兼容组件 HttpBrowserCapabilities
<Compilation>	设定 ASP.NET 应用程序的编译方式
<CustomErrors>	设定 ASP.NET 应用程序的自动错误处理
<Globalizations>	关于 ASP.NET 应用程序的全球化设定，也就是本地化设定
<HttpHandlers>	设定 HTTP 处理是对应到 URL 请求的 HttpHandler 类
<HttpModules>	创建\删除或清除 ASP.NET 应用程序的 HTTP 模块
<HttpRuntime>	设定 ASP.NET HTTP 运行时的配置，这些配置决定如何处理对 ASP.NET 应用程序的请求
<MachineKey>	设定在使用窗体基础验证的 Cookie 数据时，用来加码和解码的金钥匙
<Membership>	设定 ASP.NET 的 Membership 机制
<Pages>	设定 ASP.NET 程序的相关设定，即 Page 指引命令的属性
<Profile>	设定个人化信息的 Profile 对象
<Roles>	设定 ASP.NET 的角色管理
<SessionState>	设定 ASP.NET 应用程序的 Session 状态 HttpModule

(续表)

设 定 区 段	说　　明
<SiteMap>	设定 ASP.NET 网站导航系统
<WebParts>	设定 ASP.NET 应用程序的网页组件
<WebServices>	设定 ASP.NET 的 Web 服务

3. 在 Web.config 文件中添加用户自定义变量

在 Web 配置文件的<appSettings>区段通过<add>标记创建的参数，在程序中可以使用 System.Web.Configuration 命名空间的 WebConfigurationManager 类来获取。

【例 2-9】读取 Web.config 中的用户变量。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 2-9】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动 1 个 Label 控件到 Web 窗体中。

(3) 打开 Web.config 文件，在<appSettings>配置子节点中添加一个变量，代码如下：

```
<appSettings>
<add key="MyName" value="葛萌萌"/>
</appSettings>
```

(4) 在 Default.aspx.cs 中添加页面的 Load 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "变量 Name 的值为：" + WebConfigurationManager.AppSettings["MyName"];
}
```

提示

在添加上述代码之前，需要先导入所需的命名空间：using System.Web.Configuration。

(5) 编译并运行程序，显示效果如图 2-24 所示。

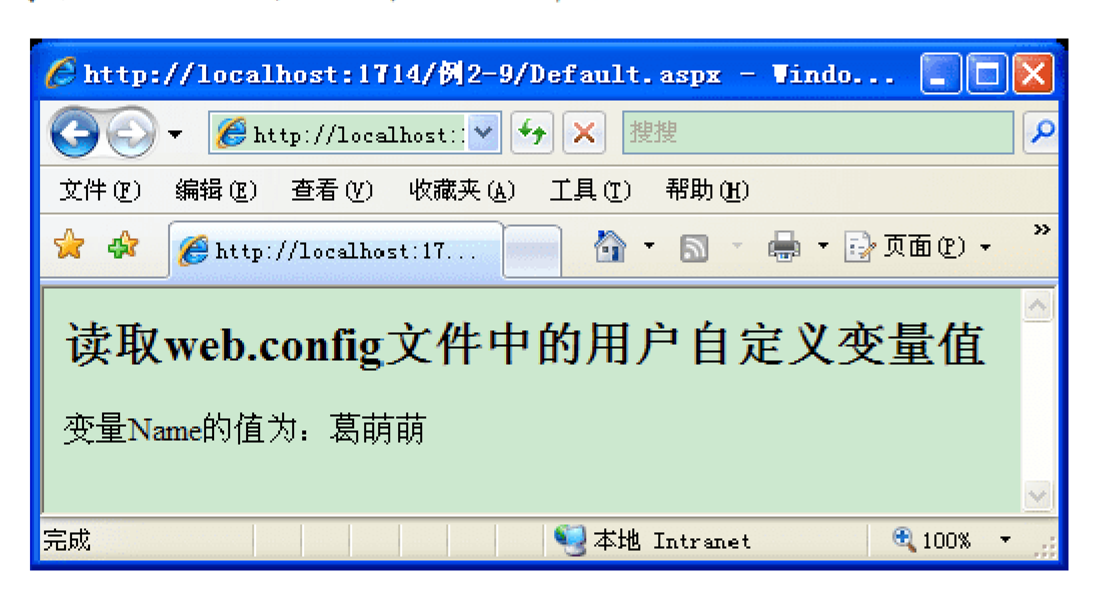


图 2-24　页面运行效果





4. 在 sessionState 区段设置 Session 状态

ASP.NET 的 Session 状态管理拥有扩展性，可以在 Web.config 文件的<sessionState>区段设定 Session 状态管理，该区段属于<system.web>子标记。

<sessionState>区段的常用属性如表 2-9 所示。

表 2-9　<sessionState>区段的常用属性

属　性	说　明
mode	Session 状态存储的模式，可以是 off(不存储)、InProc(使用 Cookie)、StateServer(使用状态服务器)和 SqlServer(存储在 SQL Server 中)
cookieless	是否使用 Cookie 存储 Session 状态。True 表示不使用，False 表示使用
timeout	Session 时间的期限，以分钟计，默认为 20 分钟，与 Session 对象的 TimeOut 属性功能相同

2.4.2　Global.asax 文件介绍



Global.asax 文件是 Web 应用程序的系统文件，属于选项文件，可有可无。当需要使用 Application 和 Session 对象的事件处理程序时，就需要创建此文件。

另外，由于 Global.asax 在网络应用程序中的特殊地位，它被存放的位置也是固定的，必须存放在当前应用所在的虚拟目录的根目录下。如果放在虚拟目录的子目录中，Global.asax 文件将不会起任何作用。



知识点

作为网络应用程序，程序在执行之前有时需要初始化一些重要的变量，而且这些工作必须在所有程序执行之前完成， Global.asax 文件便是为此目的而设计的。

Global.asax 是 ASP.NET 应用程序的“全局应用程序类”，该文件是应用程序用来保持应用程序级的事件、对象和变量的。一个 ASP.NET 应用程序只能有一个 Global.asax 文件。

按照 VWD 模板添加的 Global.asax 如下所示。

```
<%@ Application Language="C#"%>
<script runat="server">
    void Application_Start(object sender,EventArgs e)
    {
        //在应用程序启动时运行的代码
    }
    void Application_End(object sender, EventArgs e)
    {
```

```
            //在应用程序关闭时运行的代码
        }
        void Application_Error(object sender, EventArgs e)
        {
            //在出现未处理的错误时运行的代码
        }
        void Session_Start(object sender, EventArgs e)
        {
            //在新会话启动时运行的代码
        }
    </script>
```

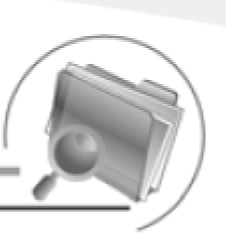
在窗体页中，只能处理单个页面的事件，而在 Global.asax 文件中则可以处理整个应用程序中的事件。除了上述代码模板中列举的事件，在 Global.asax 文件中还可以加入其他事件的处理函数。如表 2-9 所示列出了可以在 Global.asax 中处理的事件。

表 2-9　可在 Global.asax 中处理的事件

事　件	说　明
Application_AuthenticateRequest	每个请求都会触发该事件，并且可以在此函数中设置自定义的验证
Application_BeginRequest	虽然在 VWD 的代码模板中没有该事件的处理，不过可以在 Global.asax 中添加。该事件是在每个请求到达服务器后且在处理该请求前触发
Application_End	应用程序关闭时触发该事件。该函数很少使用，因为 ASP.NET 可以很好地关闭和清除内存对象
Application_Error	在应用程序中抛出任何错误时都会触发该事件。通常在此函数中提供应用程序级的错误处理或者记录错误事件
Application_Start	在应用程序接收到第一个请求时调用，通常在此函数中定义应用程序级变量或状态
Session_Start	类似于 Application_Start，不过是针对每个客户端第一次访问应用程序时调用
Session_End	以进程内模式使用会话状态时，如果用户离开应用程序将会触发该事件

与页面指令一样，Global.asax 文件也可以使用应用程序指令，这些指令都可以包含特定于该指令的一个或多个属性/值对。下面列出了 ASP.NET 中支持的应用程序指令。

- @Application：定义 ASP.NET 应用程序编译器所使用的应用程序特定的属性。该指令只能在 Global.asax 文件中使用。
- @Import：显式将命名空间导入到应用程序中。
- @Assembly：在分析时将程序集链接到应用程序。



2.5　上机练习

本章的上机练习将演示 Global.asax 文件的使用，同时回顾前面介绍的内置对象的使用。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 2】，单击【确定】按钮。

(2) 在 Global.asax 文件中添加如下代码：

```
<script runat="server">
    void Application_Start(object sender, EventArgs e)
    {
        // 在应用程序启动时运行的代码
        Application["info"] = "Application  开始...<br/>"
    }
    void Application_End(object sender, EventArgs e)
    {
        //   在应用程序关闭时运行的代码
    }
    void Application_Error(object sender, EventArgs e)
    {
        // 在出现未处理的错误时运行的代码
    }
    void Session_Start(object sender, EventArgs e)
    {
        // 在新会话启动时运行的代码
        Response.Write(Application["info"].ToString());
        Application["info"] = "";//清空 Application 变量
        Response.Write("Session  开始...<br/>");
    }
    void Session_End(object sender, EventArgs e)
    {
        Application["info"] = "Session  结束...<br/>";
    }
    void Application_BeginRequest(object sender, EventArgs e)
    {
        Response.Write("Request  开始...<br/>");
    }
    void Application_EndRequest(object sender, EventArgs e)
    {
        Response.Write("Request  结束...<br/>");
    }
</script>
```



(3) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动两个 Button 控件到 Web 窗体中，其 Text 属性分别为“刷新”和“结束会话”。

(4) 在 Default.aspx 页面的<body>标记中添加如下代码:

```
<body>
    页面内容...
    (<% if (Session.IsNewSession)
            Response.Write("新的 Session 时间");
        else
            Response.Write("同一个 Session 时间");
     %>)
    <form id="form1" runat="server">
    <div>
        <asp:Button ID="Button1" runat="server" Text="刷新" onclick="Button1_Click" /> 
        <asp:Button ID="Button2" runat="server" Text="结束会话" onclick="Button2_Click" />
    </div>
    </form>
</body>
```

(5) 在 Default.aspx.cs 中添加页面的 Load 事件和两个按钮的单击事件处理程序，代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("加载页面...<br/>");
}
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("刷新页面...<br/>");
}
protected void Button2_Click(object sender, EventArgs e)
{
    Session.Abandon();//结束 Session
    Response.Redirect("Default.aspx");
}
```

(6) 编译并运行程序，启动默认浏览器打开 Default.aspx 文件，效果如图 2-25 所示。从运行结果可以看出事件处理程序的执行顺序。

(7) 单击【刷新】按钮，由于此时 Session 尚未结束，所以页面内容显示的是“同一个 Session 时间…”，如图 2-26 所示。



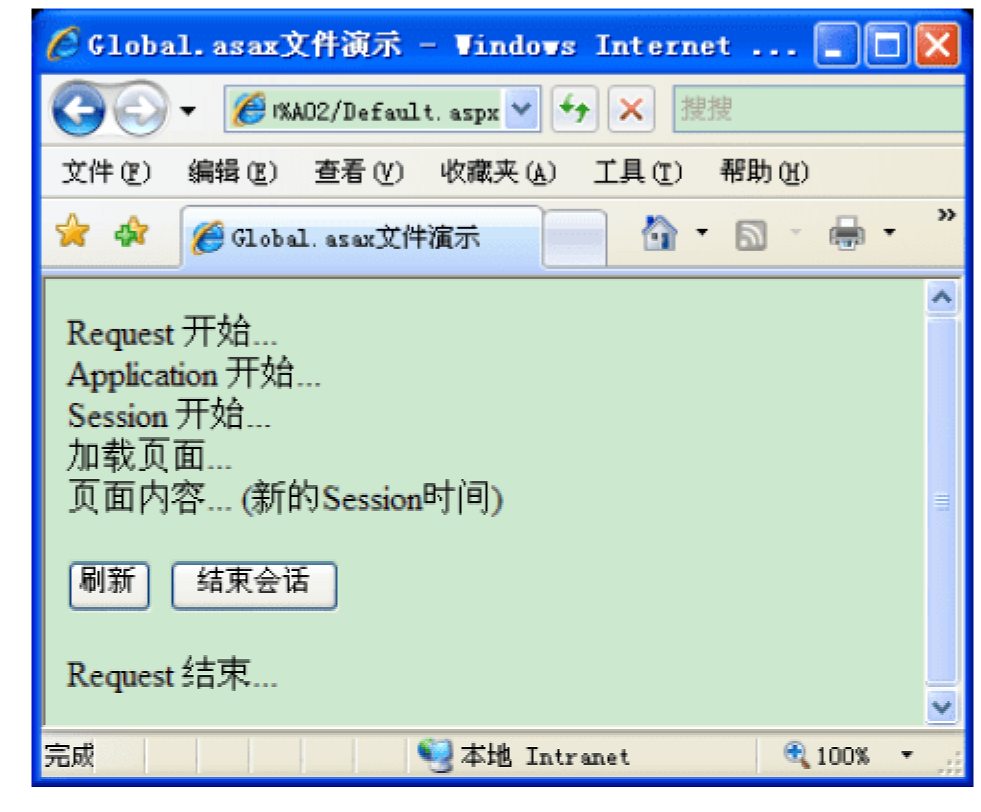


图 2-25　页面初次加载效果

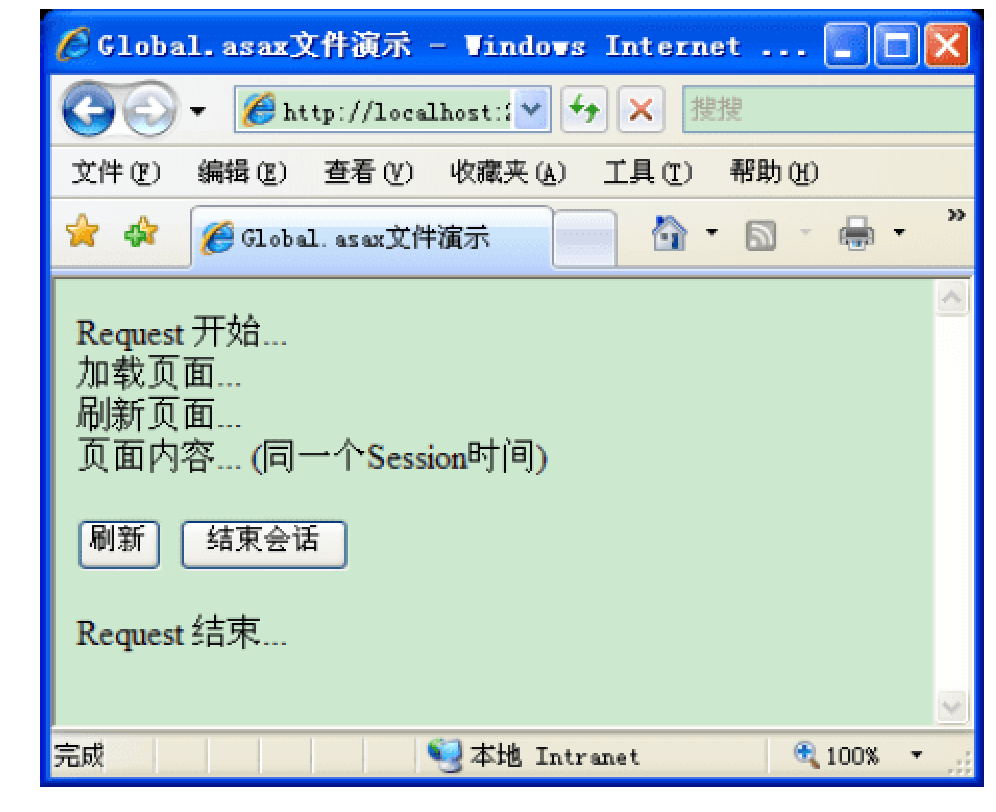


图 2-26　刷新页面

(8) 单击【结束会话】按钮，因为需要在程序中以 Abandon()方法强制结束 Session 事件，然后重新加载页面，可以看到，此次显示为新的 Session 时间，如图 2-27 所示。

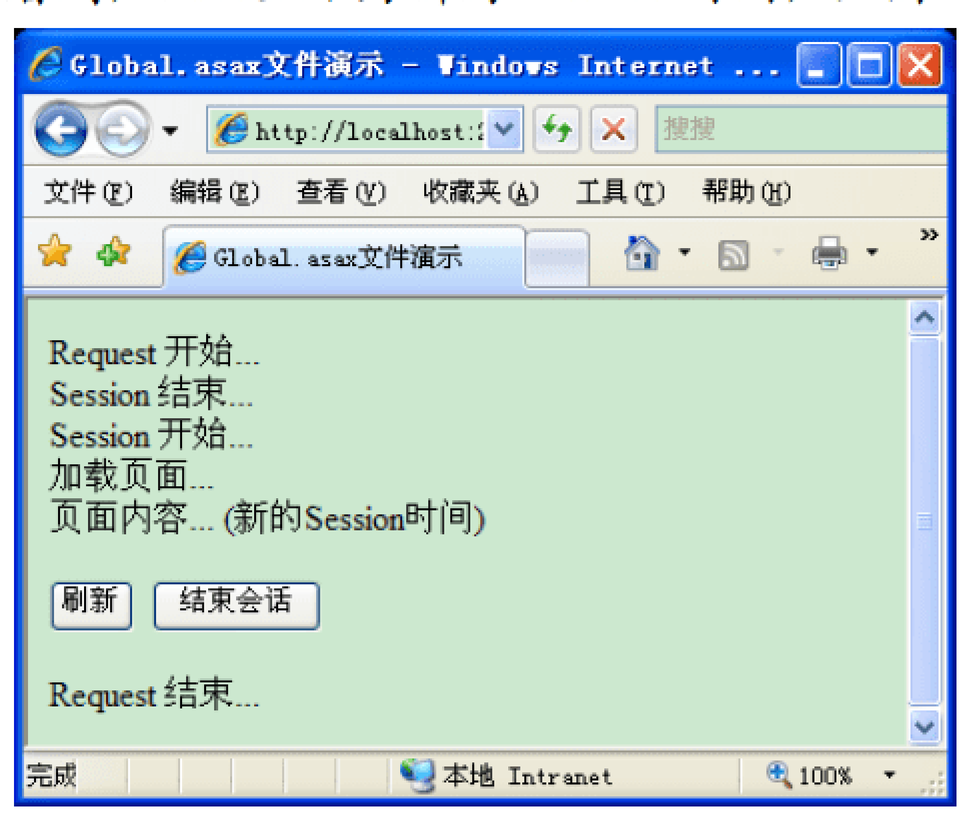


图 2-27　结束会话重新加载页面

2.6　习题

1. App_Data 目录主要存放什么文件？

2. App_Code 目录和 bin 目录有什么区别？

3. 简述加载页面时，Page 类各事件的发生顺序。

4. 新建一个网站，通过 Application 对象统计网站的访问人数。

5. 新建一个 ASP.NET 页面，分别使用 Response.Redirect()和 Server.Transfer()方法跳转到清华大学出版社的网站：http://www.tup.com.cn。

6. 对于 HTML 表单，使用 Get 方法或 Post 方法提交数据有什么区别？

7. 新建一个 ASP.NET 页面，使用 Request 对象的 Browser 属性来获取客户端浏览器信息，



包括浏览器类型、浏览器版本信息、浏览器使用的平台等。

8. 新建一个 ASP.NET 页面，使用 Server 对象获取服务器端的信息，包括服务器名称、超时时间和文件的物理路径等。

9. Web.config 文件是什么格式的？该配置文件包含哪些配置的设置。

10. Global.asax 文件的作用是什么？



ASP.NET服务器控件

学习目标

ASP.NET 服务器控件是 ASP.NET 网页上的对象。使用 ASP.NET 服务器控件，可以大大减少开发 Web 应用程序所需编写的代码量，提高开发效率和 Web 应用程序的性能。ASP.NET 服务器控件的体系结构已经完全集成到了 ASP.NET 中，为用户提供了一个构建 Web 站点的技术中相当独特的功能集。本章将介绍这些服务器控件的基本用法以及不同类别控件的功能。这些控件在每个 ASP.NET 应用程序中都会用到。因此，了解工具箱中有哪些控件可用、它们各自的用途、它们的工作原理以及它们如何维持自身状态非常关键。

本章重点

- ASP.NET 服务器控件的概念和工作原理
- 服务器控件的基类和常用事件
- 列表控件的使用
- 各种验证控件的功能和用法
- 使用 ASP.NET 的导航控件
- 使用登录控件
- 如何创建和使用用户控件
- ASP.NET 状态引擎

3.1 ASP.NET 服务器控件概述

ASP.NET 服务器控件是 ASP.NET 的重要组成部分。在 VWD 中构建的几乎所有页面都包含一个或多个服务器控件。这些控件可分为不同的类型，有 Button 和 Label 这样的简单控件，也有复杂的控件，如可以显示数据源中数据的控件 TreeView 和 GridView。

ASP.NET 服务器控件是服务器端 ASP.NET 网页上的对象，当用户通过浏览器请求 ASP.NET 网页时，这些控件将运行并把生成的标准的 HTML 文件发送给客户端浏览器来呈现。

在 ASP.NET 页面上，服务器控件表现为一个标记，例如<asp:TextBox…/>。这些标记不是标准的 HTML 元素，因此如果它们出现在网页上，浏览器将无法识别它们，然而，当从 Web 服务器上请求一个 ASP.NET 页面时，这些标记都将动态地转换为 HTML 元素。

3.1.1 服务器控件类

大多数 Web 服务器控件类都派生于 System.Web.UI.WebControls.WebControl 类，而 WebControl 类又从 System.Web.UI.Control 类派生而来。

System.Web.UI.WebControls 命名空间中的服务器控件可分为以下两类。

- Web 控件：用来构建与用户进行交互的页面。这类控件包括常用的按钮控件、文本框控件、复选框控件以及用户自定义控件等，使用这些控件可以创建与用户交互的接口。
- 数据绑定控件：用来实现数据的绑定与显示。这类控件包括广告控件、表格控件以及用于导航的菜单控件和树形控件等。

1. 基本属性

WebControl 类是用作定义 System.Web.UI.WebControls 命名空间中的所有控件的公共方法、属性和事件的基类，其中定义了一些可以应用于几乎所有服务器控件的基本属性，如表 3-1 所示。

表 3-1 WebControl 类的基本属性

属 性	说 明
AccessKey	允许设置一个键，使用这个键，就可以按关联的字母键在客户端访问控件
BackColor	获取或设置 Web 服务器控件的背景色
BorderColor	获取或设置 Web 服务器控件的边框颜色
BorderStyle	获取或设置 Web 服务器控件的边框样式
BorderWidth	获取或设置 Web 服务器控件的边框宽度
CssClass	获取或设置 Web 服务器控件在客户端呈现的级联样式表(CSS)类
Enabled	获取或设置是否启用 Web 服务器控件，默认值为 True
EnableTheming	获取或设置是否对 Web 服务器控件应用主题
Font	获取或设置 Web 服务器控件关联的字体属性
ForeColor	获取或设置 Web 服务器控件的前景色，通常为文本颜色
Height	获取或设置 Web 服务器控件的高度
ID	获取或设置 Web 服务器控件的编号标识符
SkinID	获取或设置应用于 Web 服务器控件的外观
Style	获取或设置将在 Web 服务器控件的外部标记上呈现的样式属性的文本属性的集合





(续表)

属　性	说　明
TabIndex	设置客户端 HTML tabindex 特性，确定用户按 Tab 键时焦点沿着页面中控件移动的顺序
ToolTip	允许设置浏览器中控件的工具提示。这个工具提示在 HTML 中被呈现为 title 特性，当用户把鼠标悬停在相关的 HTML 元素上时就会显示出来
Visible	获取或设置 Web 服务器控件是否作为 UI 呈现在页面上
Runat	该属性设置为 Server 时，表示该控件是一个服务器控件
Width	获取或设置 Web 服务器控件的宽度

2. 颜色与字体

在.NET 框架中，System.Drawing 命名空间提供了一个 Color 对象，使用该对象可以设置控件的颜色属性。创建颜色的方式有以下 3 种。

- 使用 ARGB(alpha,red,green,blue)颜色值：可以为每个值指定一个 0~255 之间的整数。其中，alpha 表示颜色的透明度，当 alpha 为 255 时，表示完全不透明；red 表示红色；green 表示绿色；blue 表示蓝色。
- 使用颜色的枚举值：可供使用的颜色有 140 个。
- 使用 HTML 颜色名：可以使用 ColorTranslator 类把字符串转换为颜色值。



例如，下面的代码都是设置控件 Button1 控件的颜色：

```
Button1.BackColor = Color.FromArgb(255,0,255,127);
Button1.BackColor = Color.DarkGreen;
Button1.BackColor = ColorTranslator.FromHtml("Green");
```

控件的字体属性依赖于 System.Web.UI.WebControls 命名空间中的 FontInfo 对象。该对象提供的属性如表 3-2 所示。

表 3-2　FontInfo 对象的属性

属　性	说　明
Name	指明字体的名称，如 Arial
Names	指明一系列字体，浏览器会首先选用第一个去匹配用户安装的字体
Size	字体的大小，可以设置为相对值或者真实值
Bold、Italic、Strikeout、Underline、Overline	布尔属性，用来设置是否应用给定的样式特征。Bold 是粗体，Italic 为斜体，Strikeout 为中划线，Underline 为下划线，Overline 为上划线

3. 服务器控件的事件

在 ASP.NET 页面中，用户与服务器的交互是通过 Web 控件的事件来完成的。例如，当单击一个按钮时，就会触发按钮的单击事件，程序员只需在该单击事件处理程序中提供相应的代码，

即可对用户的单击行为作出响应。

Web 控件的事件工作方式与传统的 HTML 标记的客户端事件工作方式有所不同，这是因为 HTML 标记的客户端事件是在客户端触发并处理的，而 ASP.NET 中的 Web 控件的事件虽然也是在客户端触发，但却是在服务器端处理的。

Web 控件的事件模型为：客户端捕捉到事件信息，接着通过 HTTP POST 将事件信息发送到服务器，而且页面框架必须解释该 POST 以确定所发生的事件，然后在要处理该事件的服务器上调用代码中的相应方法。

基于以上事件模型为，Web 控件事件可能会影响到页面的性能，因此，Web 控件仅仅提供有限的一组事件，如表 3-3 所示。

表 3-3　Web 控件的事件

事　　件	支持的控件	功　　能
Click	Button、ImageButton	单击事件
TextChanged	TextBox	输入焦点变化
SelectedIndexChanged	DropDownList、ListBox、CheckBoxList、RadioButtonList	选择项变化

Web 控件通常不再支持经常发生的事件，如 OnMouseOver 事件等，因为如果在服务器端处理这些事件，就会浪费大量的资源，但 Web 控件仍然可以为这些事件调用客户端处理程序。此外，控件和页面本身在每个处理步骤都会触发生命周期事件，如 Init 事件、Load 事件和 PreRender 事件，在应用程序中可以使用这些生命周期事件。

所有的 Web 事件处理函数都包括两个参数：第一个参数表示触发事件的对象，第二个参数表示包含该事件特定信息的事件对象，通常是 EventArgs 类型，或 EventArgs 类型的子类型。例如，按钮控件的单击事件处理程序，其代码形式如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    //在此添加处理程序
}
```

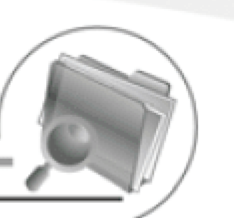
4. 事件的绑定

在处理 Web 控件时，经常需要把事件绑定到事件处理程序。将事件绑定到事件处理程序的方法有如下两种。

(1) 在 ASP.NET 页面中，在声明控件时，指定该控件的事件对应的事件处理程序。例如，把一个 Button 控件的 Click 事件绑定到名为 MyClick 的方法，代码如下：

```
<asp:Button ID="Button1" runat="server" Text="Button" OnClick="MyClick" />
```

(2) 如果控件是动态创建的，则需要通过编写代码动态地将事件绑定到方法，例如：



```
Button myBtn = new Button("Button1");

myBtn.Text = "提交";
myBtn.Click += new System.EventHandler(ButtonClick);
```



提示

在以定义的方式把事件绑定到事件处理程序时，在控件定义标记中都以 On 开头跟着事件名称的形式出现，如上面的 onclick。

3.1.2 使用服务器控件

在构建 ASP.NET Web 页面时会用到大量的服务器控件，因此需要详细了解它们的工作原理以及如何使用它们。在 ASP.NET 中，Web 控件具有特殊的格式，总是以 asp:前缀开始，后跟控件类的类名。如果该 Web 控件不需要一个关闭标记，则 Web 控件的标记必须以/>结束。

在前面的学习中，已经使用过 Label、Button 和 TextBox 控件，要向页面中添加服务器控件，只需简单地从工具箱中拖动相应的控件到设计视图中即可。

下面通过一个具体的实例来看一下服务器控件的工作原理。

【例 3-1】使用服务器控件。在本例中，将向页面添加一个 TextBox 控件、一个 Label 控件和一个 Button 控件。当客户端浏览器请求该页面时，会将这些服务器控件转换为 HTML，然后再发送 HTML 给客户端。如果在浏览器中查看页面的最终 HTML，就能发现这些 HTML 完全不同于初始的 ASP.NET 标记。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-1】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的【设计】视图，从【工具箱】中拖动 4 个 Label 控件、4 个 TextBox 控件和两个 Button 控件到 Web 窗体中，控件的 Text 属性和布局如图 3-1 所示。

(3) 选中 TextBox2 控件，在【属性】窗口中设置其 ReadOnly 属性为 True，如图 3-2 所示。

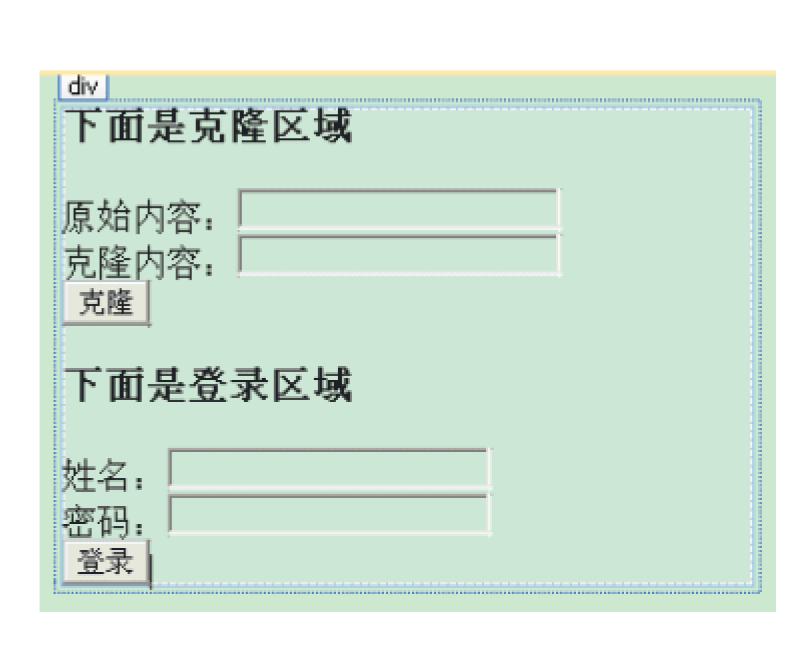


图 3-1　窗体中的控件布局

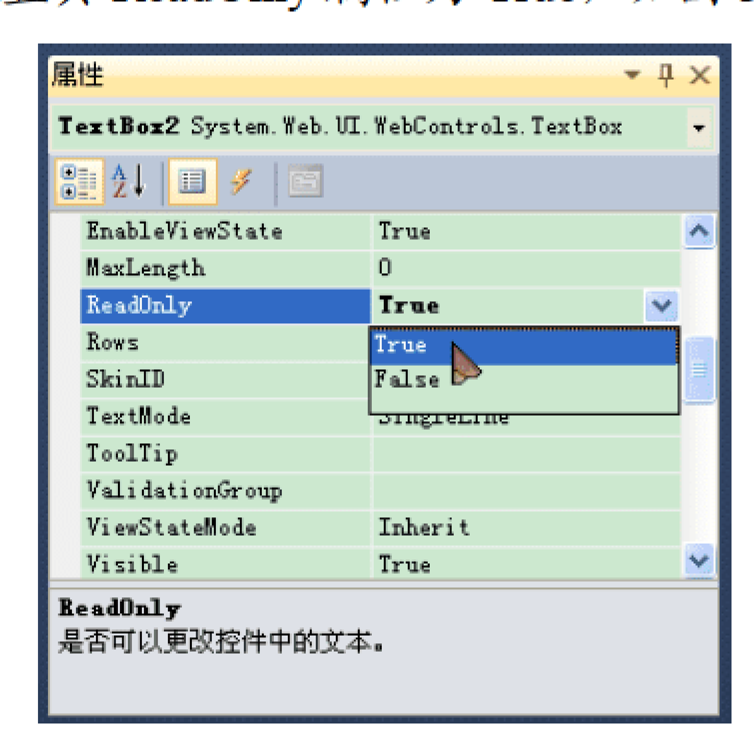


图 3-2　设置 TextBox2 的 ReadOnly 属性



(4) 双击【克隆】按钮，添加按钮的单击事件处理程序，代码如下:

```
protected void Button1_Click(object sender, EventArgs e)
{
    TextBox2.Text = Server.HtmlDecode(Server.HtmlEncode(TextBox1.Text));
}
```

(5) TextBox4 在此是作为密码输入框的，所以需要设置其 TextMode 属性为 Password，如图 3-3 所示。

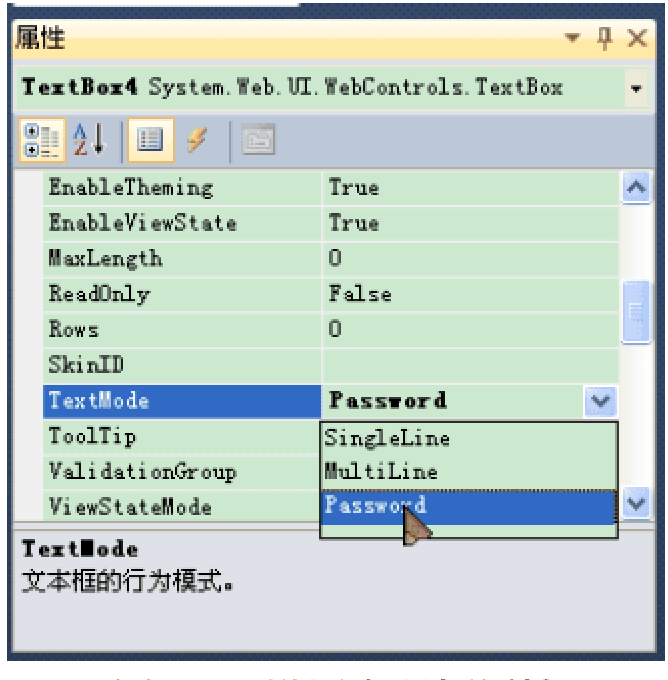


图 3-3　设置密码框属性

(6) 在设计视图中，双击【登录】按钮控件，添加控件的 Click 事件的处理方法，代码如下:

```
protected void Button2_Click(object sender, EventArgs e)
{
    if (TextBox3.Text =="" || TextBox4.Text == "")
    {
        string info = "alert(\"请输入姓名和密码！ \"); ";
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(),"warning", info, true);
    }
    else
    {
        Response.Redirect(string.Format("login.aspx?user={0}&password={1}",
        TextBox3.Text, TextBox4.Text));
    }
}
```

(7) 这段代码中，首先检查 TextBox3 和 TextBox4 中的内容是否为空，若为空，则通过 alert 语句弹出对话框，提示用户输入姓名和密码；若不为空，则将控件中的数据作为参数传给 login.aspx 页面。所以接下来还需添加 login.aspx 页面。

(8) 在【解决方案资源管理器】窗口中，右击【例3-1】解决方案，从弹出的快捷菜单中选择【添加新项】命令，在打开的【添加新项】对话框中选择【Web 窗体】，在【名称】文本框中输



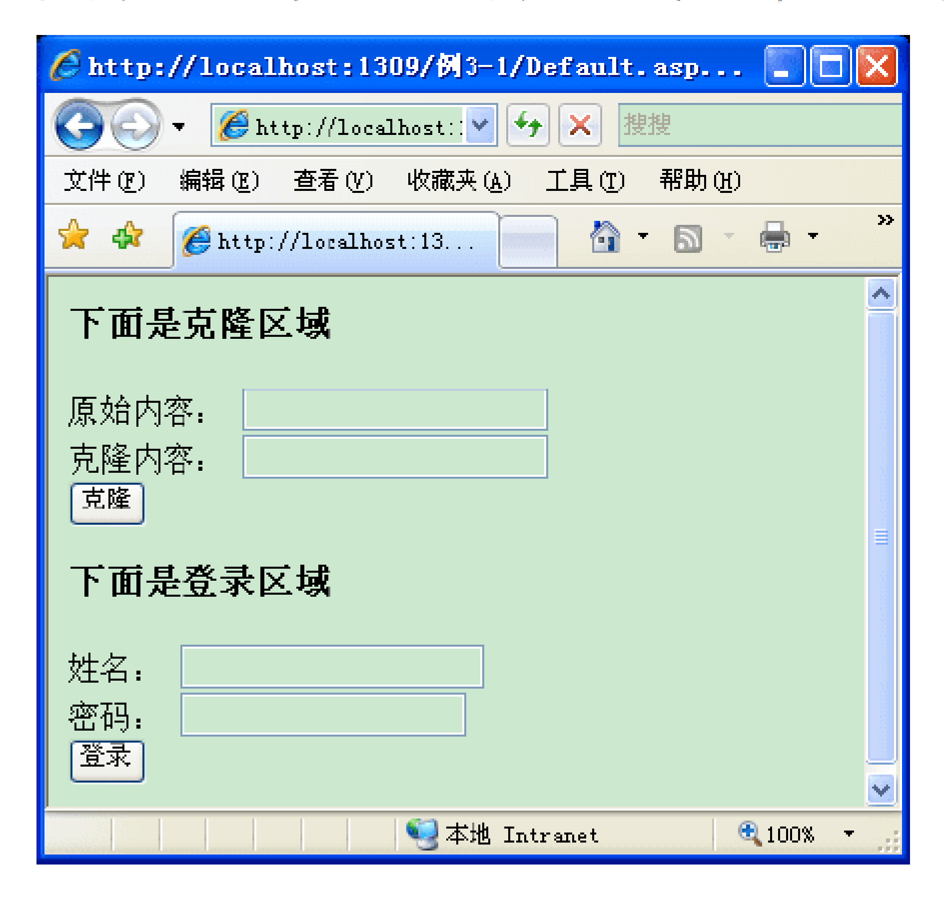
入文件名 login.aspx，然后单击【添加】按钮。

(9) 在 login.aspx 页面的 Load 事件中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("欢迎你，");
    Response.Write(Request.QueryString["user"].ToString());
    Response.Write("<br/>请牢记你的密码：");
    Response.Write(Request.QueryString["password"].ToString());
}
```

(10) 编译并运行程序，在浏览器中加载页面 Default.aspx，如图 3-4 所示，如果试图向【克隆内容】文本框中输入信息会发现无法输入，因为已将该文本框设置为只读。

(11) 在【原始内容】文本框中输入一些信息，单击【克隆】按钮，将把【原始内容】文本框中的内容克隆到【克隆内容】文本框中，如图 3-5 所示。



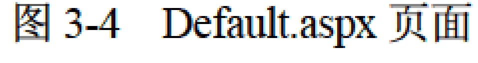


图 3-4　Default.aspx 页面

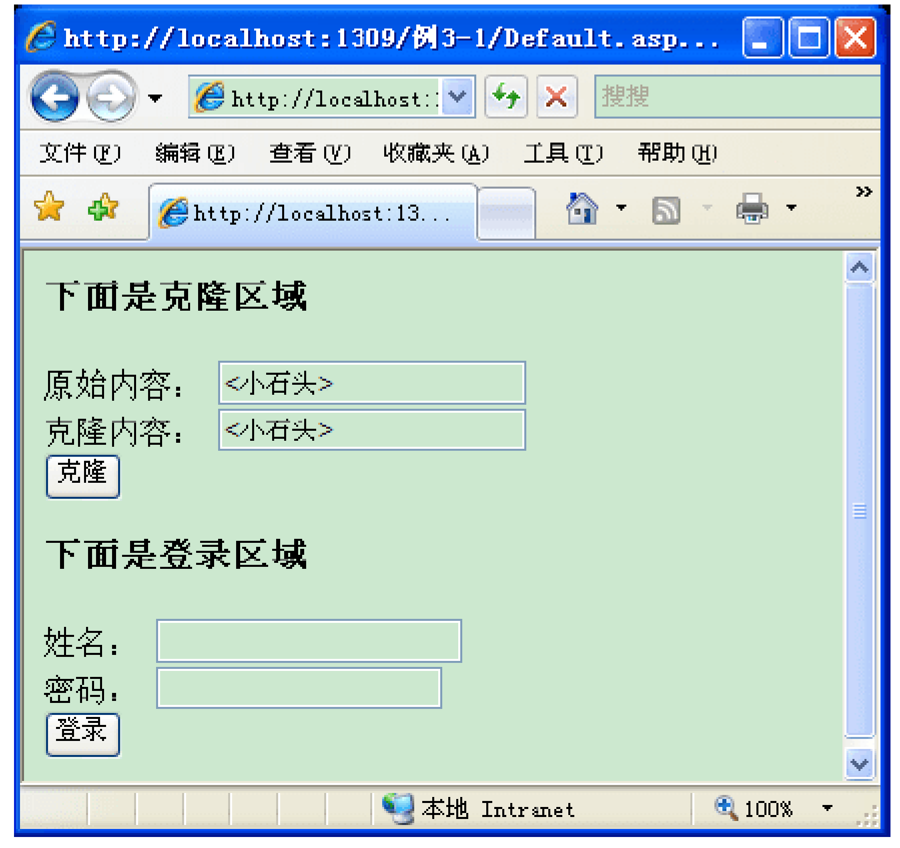


图 3-5　克隆内容

(12) 如果不输入姓名和密码就单击【登录】按钮，将弹出提示对话框，如图 3-6 所示。

(13) 输入正确的姓名和密码后，单击【登录】按钮，将跳转到 login.aspx 页面，如图 3-7 所示。



图 3-6　提示对话框

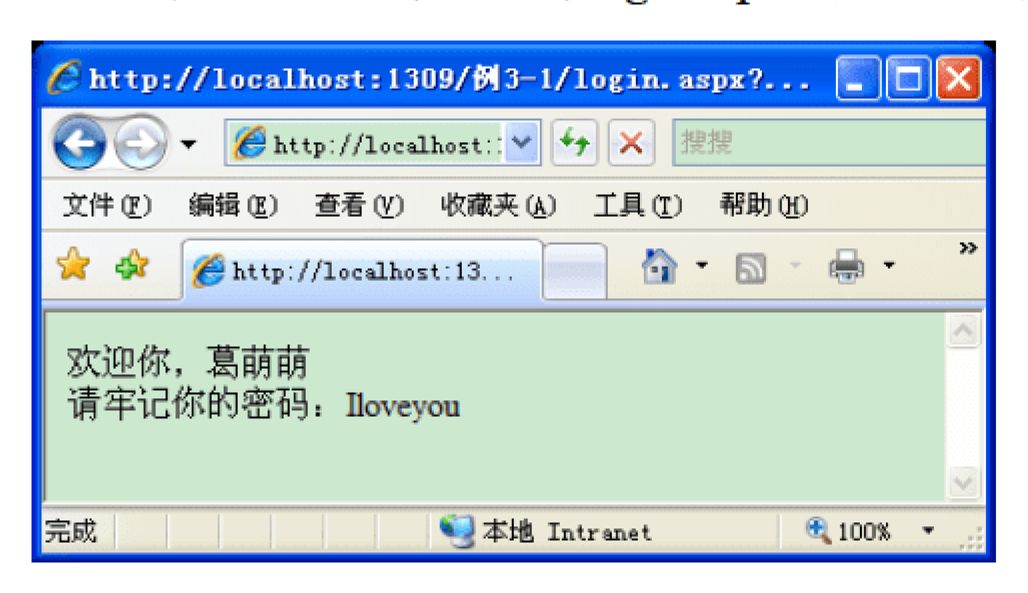


图 3-7　登录成功



下面来看一下服务器控件的工作原理。当在浏览器中请求页面时，服务器端控件就由ASP.NET 运行库(负责接收和处理 aspx 页面请求的引擎)处理，然后控件就会输出客户端 HTML 代码，并将其附加到最终页面输出的后面，最后出现在浏览器中用来构建页面的就是该 HTML 代码。例如，当首次加载 Label 控件并请求它的 HTML 时，它会返回下面的代码：

```
<span id="Label1"></span>
```

从上面这行代码可以看出，虽然使用<asp:Label>语法定义了 Label 控件，但是它最终出现在浏览器中的只是一个简单的<span>元素。因为该Label控件的Text属性是空的，所以在两个<span>标记中看不到任何内容。其他控件也是如此。

当请求【例 3-1】的 Default.aspx 页面以后，可以通过【查看】|【源文件】命令查看每个控件的 HTML 代码，如下所示：

```
<div>
    <h3>下面是克隆区域</h3>
    <span id="Label1">原始内容：</span>
    <input name="TextBox1" type="text" id="TextBox1" /><br />
    <span id="Label2">克隆内容：</span>
    <input name="TextBox2" type="text" readonly="readonly" id="TextBox2" /><br />
    <input type="submit" name="Button1" value="克隆" id="Button1" /><br />
    <h3>下面是登录区域</h3>
    <span id="Label3">姓名：</span>
    <input name="TextBox3" type="text" id="TextBox3" style="text-align: left" /><br />
    <span id="Label4">密码：</span>
    <input name="TextBox4" type="password" id="TextBox4" /><br />
    <input type="submit" name="Button2" value="登录" id="Button2" />
</div>
```

回忆一下例 3-1 中，当没有输入姓名和密码时就单击【登录】按钮，将弹出提示对话框，这个提示对话框是通过 JavaScript 的 alert 语句产生的。而在程序中是通过 Page.ClientScript.RegisterClientScriptBlock 来实现的，那么，输出到客户端以后，生成的 HTML 代码是什么呢？用户可以在源文件的<body>标记上方找到相应的代码，如下所示：

```
<script type="text/javascript">
//<![CDATA[
alert("请输入姓名和密码！"); //]]>
</script>
```

至此已经了解了服务器控件的工作原理，下一节将具体介绍 ASP.NET 4.0 附带的服务器控件的使用方法与技巧。



3.2　控件的类别

ASP.NET 4.0 本身附带了大量的服务器控件，能够满足 Web 开发的大部分需要。为了更容易地找到正确的控件，因此将它们放在工具箱的各个单独的控件类别中。如图 3-4 所示为工具箱中的所有可用控件类别。

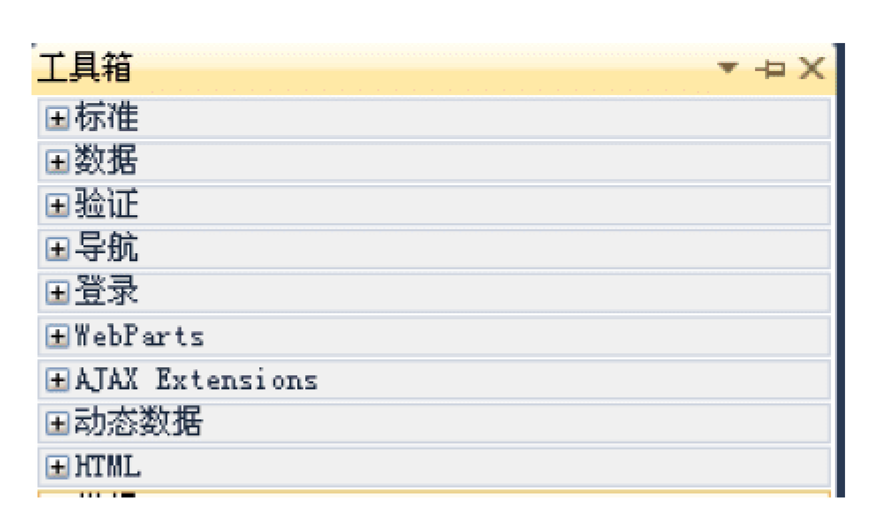


图 3-8　工具箱中的控件类别



3.2.1　标准控件

标准类别中包含很多基本控件，几乎所有的 Web 页面都需要它们。前面已经使用过其中的一部分，如 TextBox、Button 和 Label 控件。

1. 简单控件

这里所说的简单控件是指简单易懂且常用的控件，包括 TextBox、Button、Label、HyperLink、RadioButton 和 CheckBox。由于本书的其余部分会多次用到这些控件，所以在此不做详细介绍。

2. 列表控件

标准类别中有许多在浏览器中表现为列表的控件。这些控件包括 ListBox、DropDownList、CheckBoxList、RadioButtonList 和 BulletedList。要向列表中添加项，可以在控件的起始和结束标记之间定义<asp:ListItem>元素，如下面的示例所示：

```
<asp:DropDownList ID="FavoriteLanguage" runat="server">
    <asp:ListItem Value="C#">C#</asp:ListItem>
    <asp:ListItem Value="Visual Basic">Visual Basic</asp:ListItem>
    <asp:ListItem Value="CSS">CSS</asp:ListItem>
</asp:DropDownList>
```

DropDownList、RadioButtonList 控件允许用户一次只能选择一项。要以编程方式查看列表控件中当前活动和选中的项，可以查看它的 SelectedValue、SelectedItem 或 SelectedIndex 属性。SelectedValue 返回一个包含选中项的值的字符串，SelectedIndex 返回列表中项基于 0 的索引。

对于允许多重选择的控件，CheckBoxList 和 ListBox，可以在 Items 集合之间循环，并且查看选中了哪些项。在这种情况下，SelectedItem 和 SelectedValue 仅返回列表中第一个选中的项；而不是返回所有选中项。



知识点

通过设置 ListBox 控件的 SelectionMode 属性为 Multiple，可允许进行多重选择。用户可以在按住 Ctrl 或 Shift 键的同时，单击以选择多项。

BulletedList 控件不允许用户作选择，因而不支持 SelectedValue、SelectedItem 或 SelectedIndex 这些属性。

当列表控件的某个选项被选中时，该控件将引发 SelectedIndexChanged 事件。默认情况下，此事件不会导致向服务器发送页，但可通过将 AutoPostBack 属性设置为 true，强制该控件立即发送。

【例 3-2】演示列表控件的使用。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-2】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动 5 个 Label 控件、1 个 TextBox 控件、1 个 Button 控件、2 个 RadioButton 控件和 DropDownList、RadioButtonList、CheckBoxList、ListBox、BulletedList 控件各 1 个到 Web 窗体中，控件的 Text 属性和布局如图 3-9 所示。

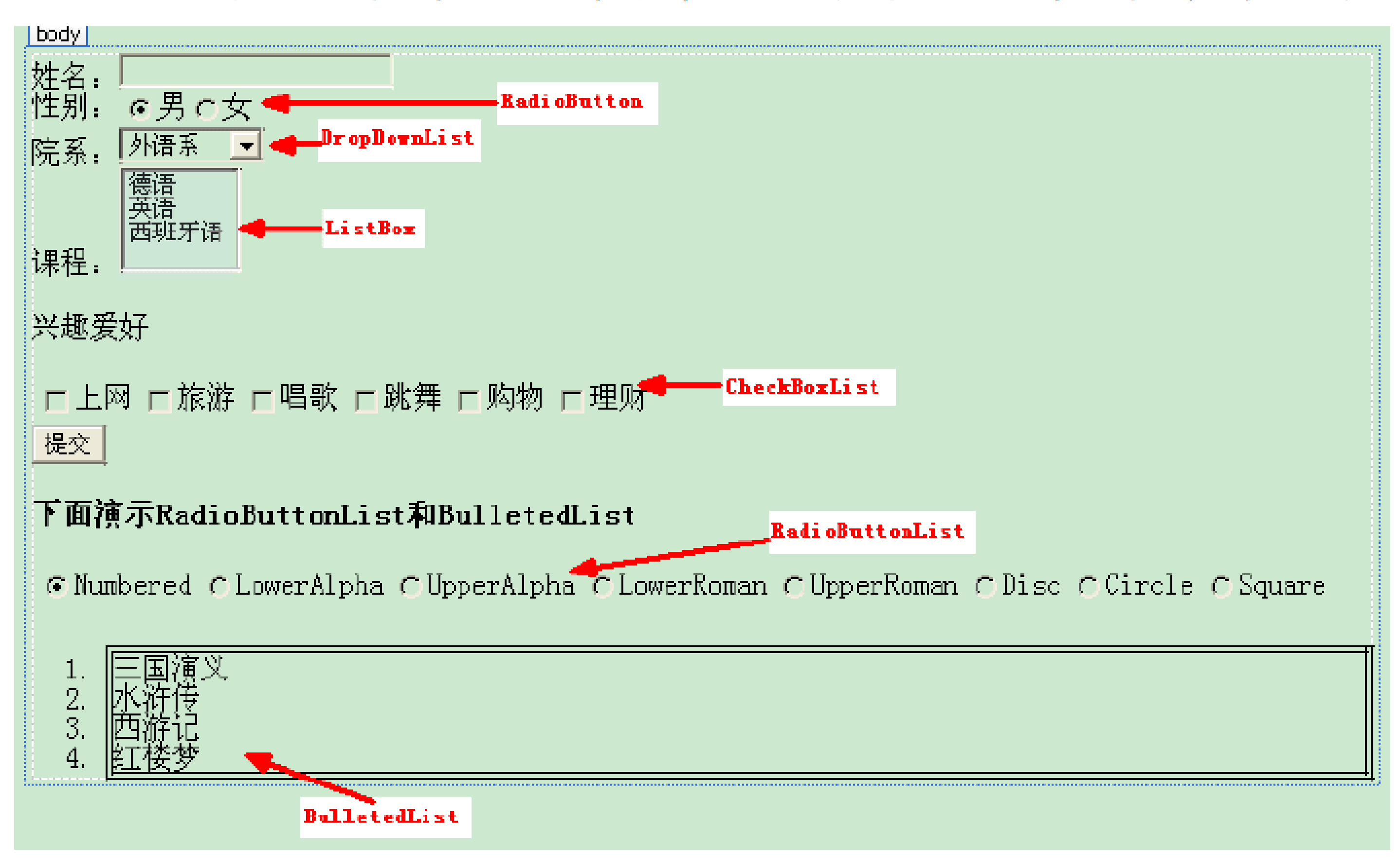
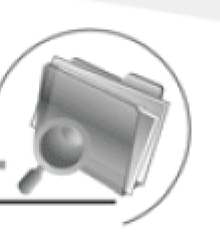


图 3-9　窗体中的控件布局



提示

在设置上述控件的属性时，2 个 RadioButton 控件的 GroupName 属性必须相同，表示这是同一组单选按钮，同一时刻只能有一个被选中。



(3) 要设置列表控件的 Items 属性，可单击【属性】面板中 Items 属性右边的按钮，如图 3-10 所示。此时将弹出【ListItem 集合编辑器】对话框，在该对话框中，单击【添加】按钮，然后在 Text 和 Value 文本框中输入相应的列表项即可，如图 3-11 所示。通过此对话框添加的项将被添加为控件标记之间的<asp:ListItem>元素。

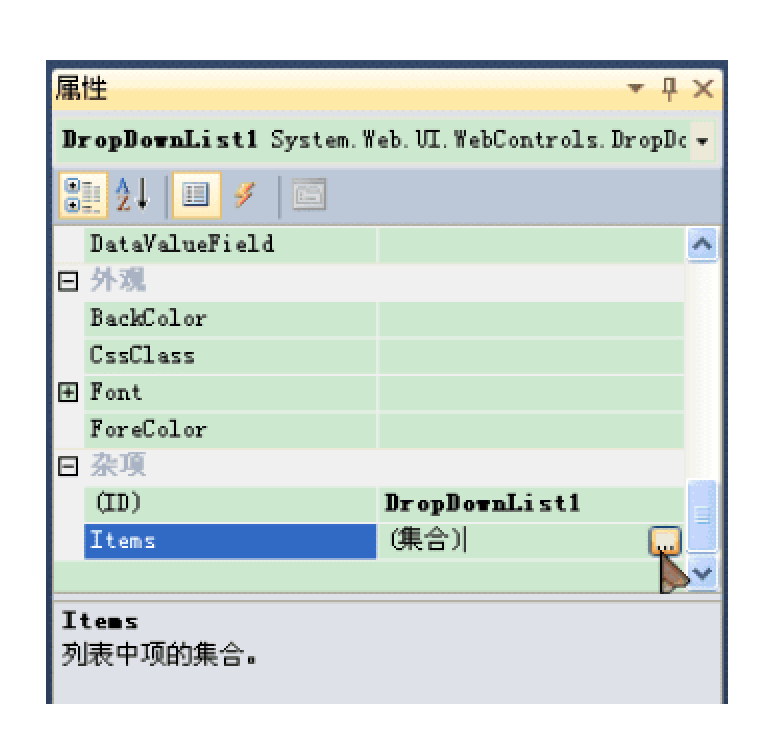


图 3-10 设置列表控件的 Items 属性



图 3-11 【ListItem 集合编辑器】对话框

(4) 设置 DropDownList 和 RadioButtonList 控件的 AutoPostBack 属性。选中 RadioButtonList 控件，单击右上角的箭头图标，打开控件的【任务】菜单，通过该菜单可以执行属于此控件的大部分常见任务。如图 3-12 所示有 3 个选项：第一个选项允许把控件与数据源绑定在一起；第二个选项用来编辑列表项，与前面设置 Items 属性相同；最后一个选项用来设置控件的 AutoPostBack 属性，选中这个选项后，一旦用户从列表中选择了一个新项，控件就会将包含它的页面提交回服务器。

(5) 为 DropDownList 和 RadioButtonList 控件添加 SelectedIndexChanged 事件处理程序。首先选中控件，然后在【属性】面板中单击工具按钮，切换到事件列表，如图 3-13 所示。在相应事件后面的文本框中双击即可添加一个默认的事件处理程序。

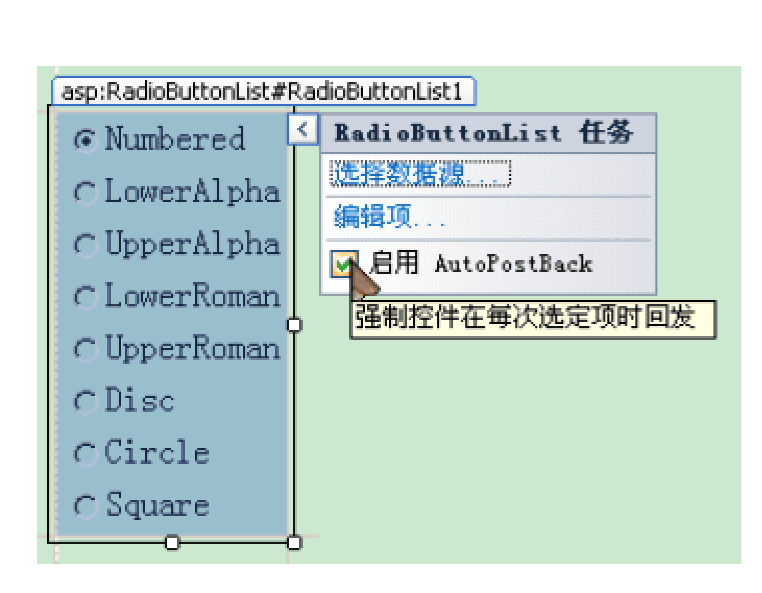


图 3-12 列表控件的【任务】菜单

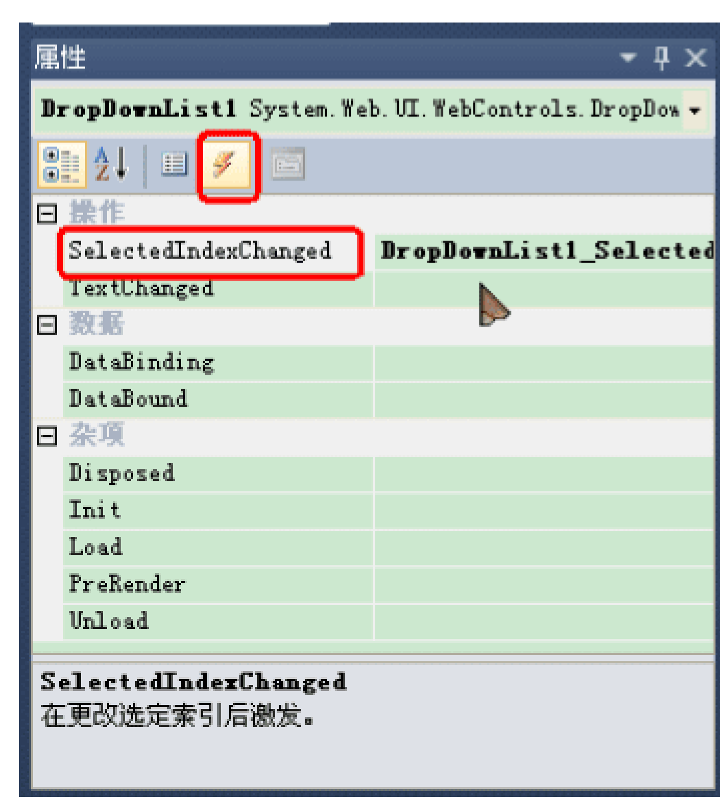


图 3-13 为控件添加事件处理程序

(6) 设置 CheckBoxList 和 RadioButtonList 控件的 RepeatDirection 属性为 Horizontal，使选项



水平排列。

(7) 设置 ListBox 控件的 SelectionMode 属性为 Multiple，以允许用户进行多项选择。

(8) 设置完控件的属性和事件后，在 Default.aspx 的源视图中可以看到以下代码:

```
<div>
    <asp:Label ID="Label4" runat="server" Text="姓名："></asp:Label>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
    <asp:Label ID="Label5" runat="server" Text="性别："></asp:Label>
    <asp:RadioButton ID="RadioButton1" runat="server" Text="男" Checked="True"
        GroupName="gender" />
    <asp:RadioButton ID="RadioButton2" runat="server" Text="女" GroupName="gender" /><br />
    <asp:Label ID="Label2" runat="server" Text="院系："></asp:Label>
    <asp:DropDownList ID="DropDownList1" runat="server"
        OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"
        AutoPostBack="True">
        <asp:ListItem>外语系</asp:ListItem>
        <asp:ListItem>数学系</asp:ListItem>
        <asp:ListItem>计算机系</asp:ListItem>
        <asp:ListItem>法律系</asp:ListItem>
    </asp:DropDownList><br />
    <asp:Label ID="Label3" runat="server" Text="课程："></asp:Label>
    <asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
        <asp:ListItem>德语</asp:ListItem>
        <asp:ListItem>英语</asp:ListItem>
        <asp:ListItem>西班牙语</asp:ListItem>
    </asp:ListBox>
    <br /><p>
    <asp:Label ID="Label1" runat="server" Text="兴趣爱好"></asp:Label>
    </p>
    <asp:CheckBoxList ID="CheckBoxList1" runat="server"
        RepeatDirection="Horizontal">
        <asp:ListItem>上网</asp:ListItem>
        <asp:ListItem>旅游</asp:ListItem>
        <asp:ListItem>唱歌</asp:ListItem>
        <asp:ListItem>跳舞</asp:ListItem>
        <asp:ListItem>购物</asp:ListItem>
        <asp:ListItem>理财</asp:ListItem>
    </asp:CheckBoxList>
    <asp:Button ID="Button1" runat="server" Text="提交" OnClick="Button1_Click" />
    <h4>下面演示 RadioButtonList 和 BulletedList</h4>
```



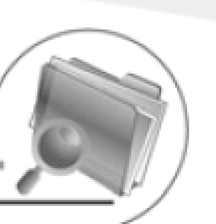
```
        <asp:RadioButtonList ID="RadioButtonList1" runat="server" AutoPostBack="True"
            onselectedindexchanged="RadioButtonList1_SelectedIndexChanged"
            RepeatDirection="Horizontal">
            <asp:ListItem Selected="True">Numbered</asp:ListItem>
            <asp:ListItem>LowerAlpha</asp:ListItem>
            <asp:ListItem>UpperAlpha</asp:ListItem>
            <asp:ListItem>LowerRoman</asp:ListItem>
            <asp:ListItem>UpperRoman</asp:ListItem>
            <asp:ListItem>Disc</asp:ListItem>
            <asp:ListItem>Circle</asp:ListItem>
            <asp:ListItem>Square</asp:ListItem>
        </asp:RadioButtonList>
        <asp:BulletedList ID="BulletedList1" runat="server" BulletStyle="Numbered"
            BorderStyle="Double" >
            <asp:ListItem>三国演义</asp:ListItem>
            <asp:ListItem>水浒传</asp:ListItem>
            <asp:ListItem>西游记</asp:ListItem>
            <asp:ListItem>红楼梦</asp:ListItem>
        </asp:BulletedList>
    </div>
```

(9) 在 Default.aspx.cs 文件中，添加控件的事件处理程序，代码如下：

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
    BulletStyle style = (BulletStyle)Enum.Parse(typeof(BulletStyle),RadioButtonList1.SelectedValue);
    BulletedList1.BulletStyle = style;
}
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    ListBox1.Items.Clear();
    string[][] strAll = new string[4][] {
        new string[] { "德语", "英语","西班牙语" },
        new string[] { "数学", "数学分析","统计学" },
        new string[] { "数据库", "网页制作" ,"ASP.NET"},
        new string[] { "刑法", "民法" ,"婚姻法"} };
    foreach(string str in strAll[DropDownList1.SelectedIndex])
    {
        ListItem item = new ListItem();
        item.Text = str;
        ListBox1.Items.Add(item);
```



```
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string strMsg = "";
        if (TextBox1.Text == "" || ListBox1.SelectedIndex < 0 || CheckBoxList1.SelectedIndex < 0)
        {
            strMsg = "alert(\"请输入姓名并选择课程和兴趣爱好！ \");";
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", strMsg, true);
        }
        else
        {
            strMsg = "alert(\"" + TextBox1.Text;
            if (RadioButton1.Checked)
                strMsg += "  先生";
            if (RadioButton2.Checked)
                strMsg += "  女士";
            strMsg += ",你好\\n 你是  ";
            strMsg += DropDownList1.SelectedValue;
            strMsg += "  的学生\\n 你选择了以下课程：";
            int i = 0;
            foreach (ListItem item in ListBox1.Items)
            {
                if (item.Selected)
                {
                    if (i++ > 0)
                        strMsg += "、";
                    strMsg += item.Value;
                }
            }
            strMsg += "\\n 你的兴趣爱好是：";
            i = 0;
            foreach (ListItem item in CheckBoxList1.Items)
            {
                if (item.Selected)
                {
                    if (i++ > 0)
                        strMsg += "、";
                    strMsg += item.Value;
                }
```



```
                }
                strMsg += "\");";
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", strMsg, true);
            }
        }
```

(10) 编译并运行程序，在浏览器中加载页面 Default.aspx，如图 3-14 所示。

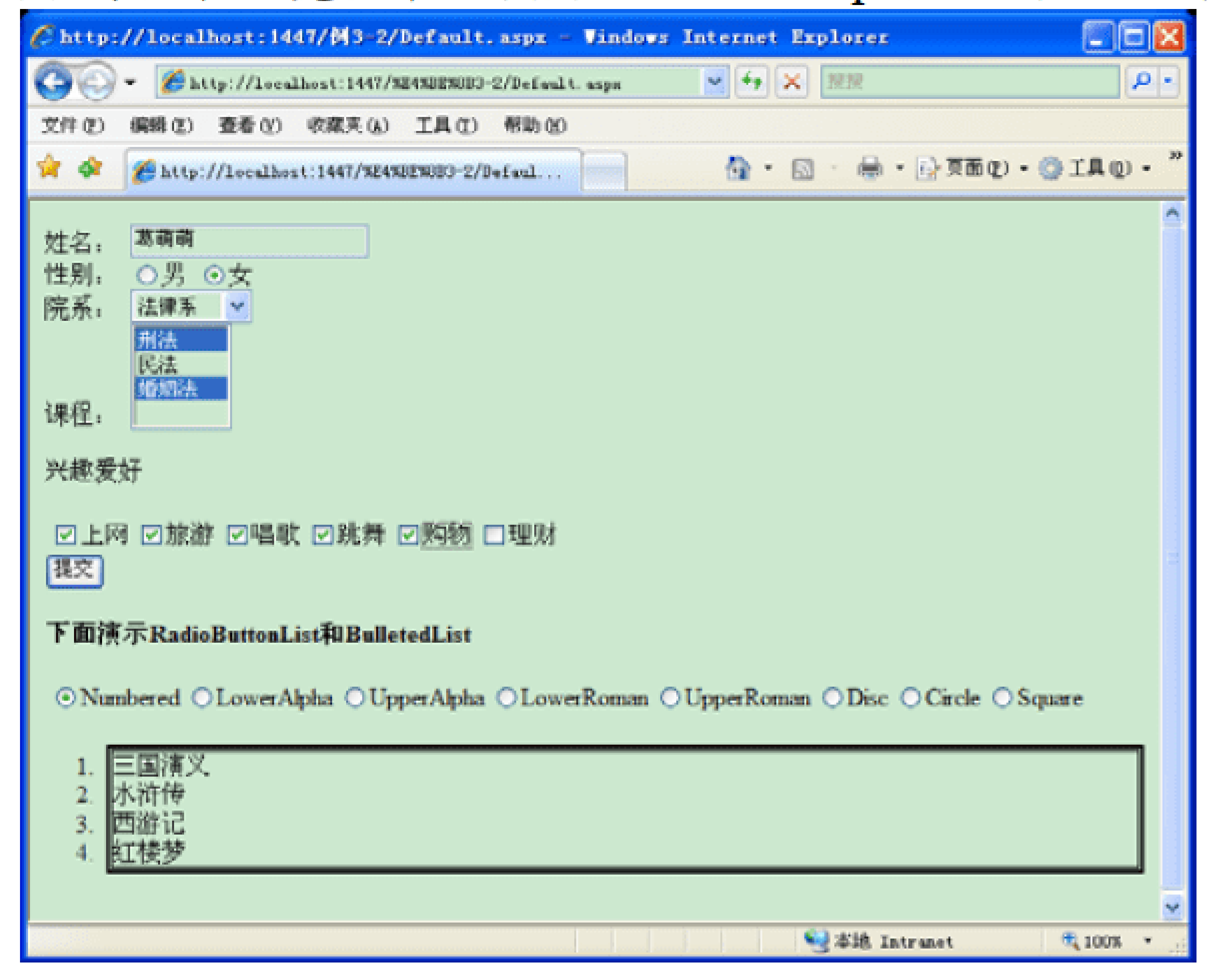


图 3-14　页面运行效果

(11) 输入姓名，选择相应的院系、课程和兴趣爱好，单击【提交】按钮，将弹出对话框，显示用户所输入的信息，如图 3-15 所示。



图 3-15　弹出对话框显示用户选择的信息



知识点

ListBox 和 CheckBoxList 控件允许用户一次性选择多个选项，因此，可使用 foreach 循环在 ListItem 元素集合中迭代逐个测试 Selected 属性。如果列表中的项为选中状态，它的 Selected 属性就为 true。

3. 容器控件

容器控件常用于以某种方式将相关的内容和控件组合到一起，常用的容器控件包括 Panel、PlaceHolder、MultiView、View 和 Wizard。例如，可以使用 PlaceHolder 或 Panel 控件同时隐藏或显示几个控件，而不用分别隐藏每个控件，只需隐藏包含各个控件和标记的整个容器即可。这两个控件各有优缺点。PlaceHolder 控件的优点是它不会向页面发布自己的 HTML，因此可以用作



容器控件，而不会在最终页面中产生任何副作用。然而，它缺少设计时支持，因此在 VWD 中难以在设计时管理 PlaceHolder 内的控件。而 Panel 控件允许轻松地访问所有控件以及它所包含的其他内容，但是它自己则呈现为<div>标记，因此，一般常使用 Panel 控件。

MultiView 和 View 控件可以制作出选项卡的效果。MultiView 控件用作一个或多个 View 控件的外部容器，View 控件可以包含标记和控件的任何组合。

如果要切换视图，可以使用控件的 ID 或者 View 控件的索引值。在 MultiView 控件中，一次只能将一个 View 控件定义为活动视图。如果某个 View 控件定义为活动视图，那么它所包含的子控件则会呈现到客户端。可以使用 ActiveViewIndex 属性或 SetActiveView 方法定义活动视图。如果 ActiveViewIndex 属性为空，则 MultiView 控件不向客户端呈现任何内容。


提示

如果活动视图设置为 MultiView 控件中不存在的 View，则会引发 ArgumentOutOfRangeException 异常。



无论是 MultiView 控件还是各个 View 控件，除当前 View 控件的内容外，都不会在页面中显示任何标记。但是，每次呈现页面时都会创建所有 View 控件中的所有服务器控件的实例，并且将这些实例的值存储为页面的视图状态的一部分。另外，可以将一个主题分配给 MultiView 或 View 控件，控件将该主题应用于当前 View 控件的所有子控件。

MultiView 和 Wizard 相似的地方是：它们允许将一个长页面划分为多个区域，这样很有好处，例如容易填写长表单。它们的区别在于 Wizard 具有使用 Previous、Next 和 Finish 按钮在页面间移动的内置支持，而 MultiView 则必须通过编程进行控制。

【例 3-3】容器控件的使用。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-3】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，从【工具箱】中拖动 1 个 CheckBox 控件和 1 个 Panel 控件到 Web 窗体中，然后在 Panel 中添加 1 个 Label 控件，切换到源视图，修改其 HTML 代码如下：

```
<asp:CheckBox ID="CheckBox1" runat="server"
    oncheckedchanged="CheckBox1_CheckedChanged" Text="显示 Panel 控件"
    AutoPostBack="True" />
<asp:Panel ID="Panel1" runat="server" Visible="False">
    <asp:Label ID="Label1" runat="server" Text="我是 Panel 控件中的 Label 控件"></asp:Label>
</asp:Panel>
```

(3) 接着输入文本信息“我要找人…”，然后添加 1 个 RadioButtonList 控件和 1 个 MultiView 控件，然后在 MultiView 控件中添加 3 个 View 控件；分别单击 3 个 View 控件输入静态文本“姓名”、“年龄”、“地域”；拖动 3 个 Textbox 控件分布到 3 个 View 控件上。然后，再拖动 1 个 Button 控件到页面上。设置 RadioButtonList 控件的 AutoPostBack 属性为 True，RepeatDirection 属性为 Horizontal，并为其添加 selectedindexchanged 事件处理程序。切换到源视图，其 HTML



代码如下：

```
<h4>我要找人...</h4>
<asp:RadioButtonList ID="RadioButtonList1" runat="server"
    RepeatDirection="Horizontal" AutoPostBack="True"
    onselectedindexchanged="RadioButtonList1_SelectedIndexChanged">
    <asp:ListItem Value="0">按姓名</asp:ListItem>
    <asp:ListItem Value="1">按年龄</asp:ListItem>
    <asp:ListItem Value="2">按地域</asp:ListItem>
</asp:RadioButtonList>
<asp:MultiView ID="MultiView1" runat="server">
<asp:View ID="View1" runat="server">
    姓名：<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</asp:View>
<asp:View ID="View2" runat="server">
    年龄：<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
</asp:View>
<asp:View ID="View3" runat="server">
    地域：<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
</asp:View>
</asp:MultiView>
<asp:Button ID="Button1" runat="server" Text="查找" onclick="Button1_Click" />
```

(4) 最后添加一个 Wizard 控件。单击控件右上角的箭头打开【Wizard 任务】面板，选择【添加/删除 WizardSteps】命令，如图 3-16 所示，将打开【WizardStep 集合编辑器】对话框，如图 3-17 所示。

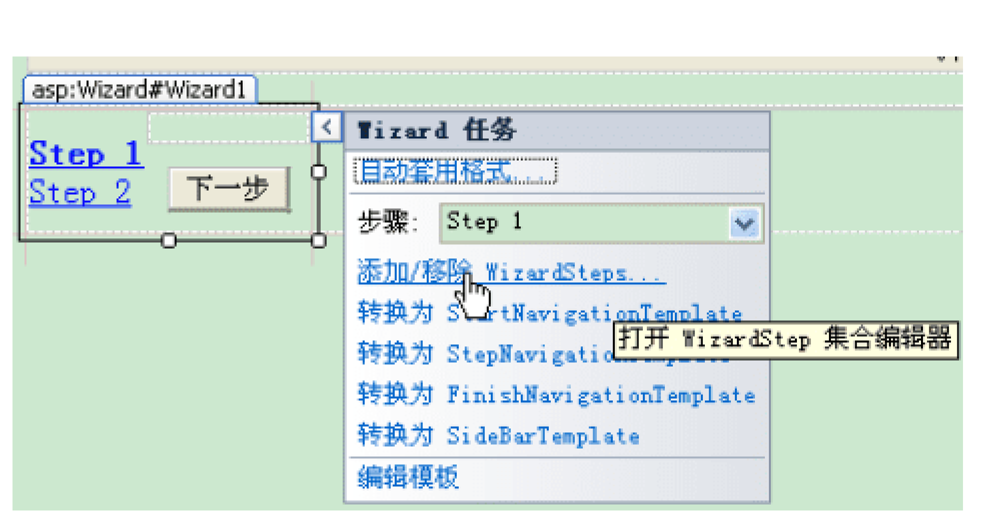


图 3-16　【Wizard 任务】面板



图 3-17　【WizardStep 集合编辑器】对话框

(5) 单击对话框左边【成员】列表框中名为 Step1 的第一个 WizardStep，将它的 Title 属性修改为"输入姓名"，将第二步的 Title 设置为"选择你喜欢的歌曲"，然后再添加一步为"完成"。

(6) 将第二步的 StepType 设置为 Finish，第三步的 StepType 设置为 Complete。可以让第一

步的 StepType 保持为 Auto。单击【确定】按钮关闭【WizardStep 集合编辑器】对话框。

(7) 在设计视图中，单击左边列表中的“输入姓名”，让它成为活动步骤，然后将一个 TextBox 控件拖到 Wizard 的右边。注意要将它拖到 Wizard 右上角的灰色矩形框内，否则控件最后不会出现在 Wizard 内。

(8) 使用同样的方法，在“选择你喜欢的歌曲”步骤中添加一个 DropDownList 控件，并为其添加若干选项，在“完成”步骤中添加一个 Label 控件。

(9) 当用户单击向导最后一步的【完成】按钮时需要进行相应的事件处理。打开控件的【属性】面板，通常默认打开【事件】选项卡。定位并双击 Action 类别中的 FinishButtonClick。切换到源视图，页面的 HTML 代码如下：

```
<asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="0"
    onfinishbuttonclick="Wizard1_FinishButtonClick">
    <WizardSteps>
        <asp:WizardStep runat="server" title="输入姓名">
            <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
        </asp:WizardStep>
        <asp:WizardStep runat="server" title="选择你喜欢的歌曲" StepType="Finish">
            <asp:DropDownList ID="DropDownList1" runat="server">
                <asp:ListItem>荷塘月色</asp:ListItem>
                <asp:ListItem>忐忑</asp:ListItem>
                <asp:ListItem>伤不起</asp:ListItem>
                <asp:ListItem>害怕爱上你</asp:ListItem>
            </asp:DropDownList>
        </asp:WizardStep>
        <asp:WizardStep runat="server" Title="完成" StepType="Complete">
            <asp:Label ID="Label2" runat="server" Text=""></asp:Label>
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>
```

(10) 此时的页面布局如图 3-18 所示。

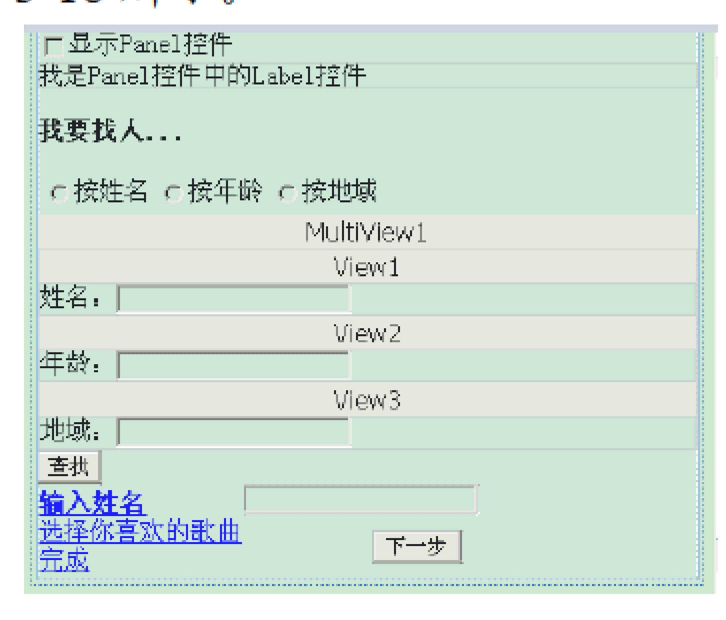
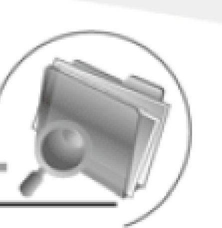


图 3-18　窗体中的页面布局


计算机基础与实训教材系列



(11) 为控件的事件添加处理代码，如下所示：

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    Panel1.Visible = CheckBox1.Checked;
}
protected void Button1_Click(object sender, EventArgs e)
{
    string strMsg = "";
    if (MultiView1.ActiveViewIndex > -1)
    {
        strMsg = "您的选择是：\\n" + RadioButtonList1.SelectedItem.Text;
        strMsg += " 查找，查找内容：";
        switch (MultiView1.ActiveViewIndex)
        {
            case 0:
                if (TextBox1.Text == "")
                    strMsg = "请输入要查找的姓名";
                else
                    strMsg += TextBox1.Text;
                break;
            case 1:
                if (TextBox2.Text == "")
                    strMsg = "请输入要查找的年龄";
                else
                    strMsg += TextBox2.Text;
                break;
            case 2:
                if (TextBox3.Text == "")
                    strMsg = "请输入要查找的地域";
                else
                    strMsg += TextBox3.Text;
                break;
            default:
                break;
        }
    }
    else
        strMsg = "请先选择找人的方式！";
    string scriptString = "alert(\"" + strMsg + "\");";
    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", scriptString, true);
```



```
}
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
    MultiView1.ActiveViewIndex = Int32.Parse(RadioButtonList1.SelectedValue);
}
protected void Wizard1_FinishButtonClick(object sender, WizardNavigationEventArgs e)
{
    Label2.Text = TextBox4.Text + " 你好：<br/>你喜欢的歌曲是：";
    Label2.Text += DropDownList1.SelectedValue;
}
```

(12) 编译并运行程序，在浏览器中加载页面 Default.aspx，如图 3-19 所示。

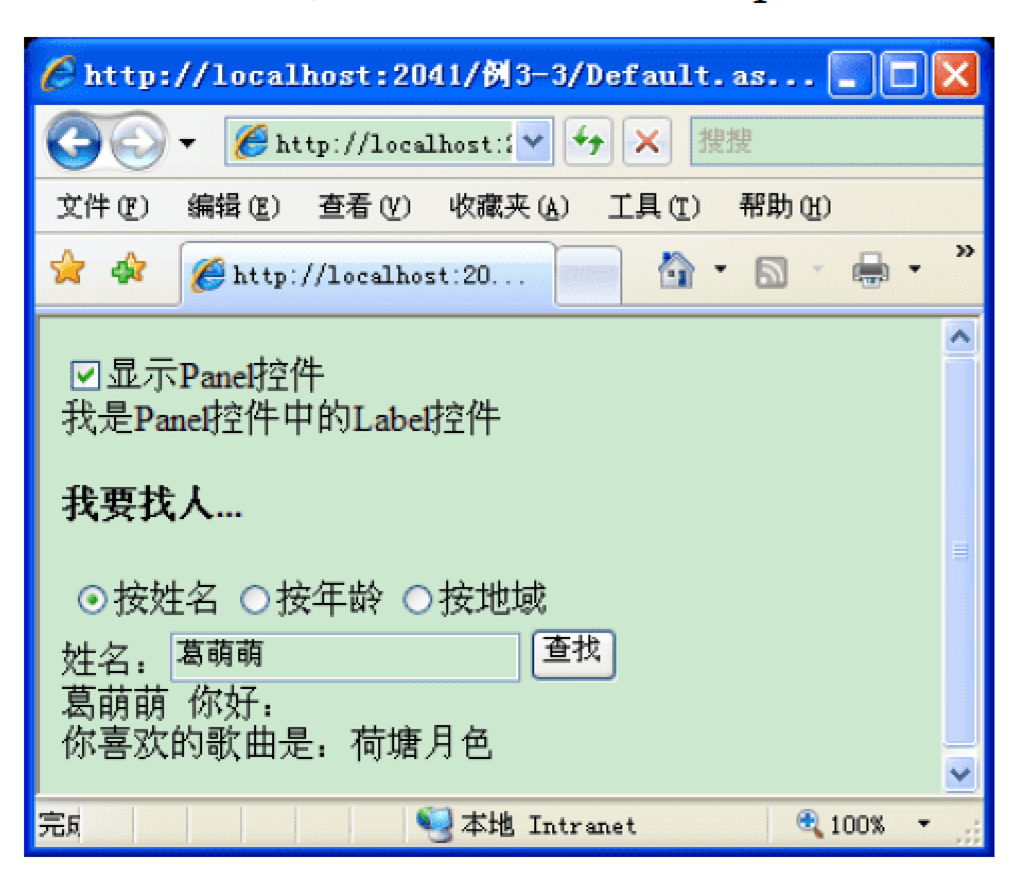


图 3-19 页面运行效果


提示

Wizard 控件能够完成大部分非常复杂的工作。它会处理导航，确定何时显示正确的按钮，并确保在结果页面中，在向导步骤中添加的控件值仍然可用，这样就可以在结果标签中显示它们。

4. 其他标准控件

除了前面介绍的简单控件、列表控件和容器控件之外，还有很多其他标准控件。它们的用法也都类似，在此只对这些控件做简单介绍。

◎ LinkButton 和 ImageButton

对 LinkButton 和 ImageButton 控件的操作类似于普通的 Button 控件。单击它们时，两者都会引起向服务器的一个回发。LinkButton 把自己显示为一个简单的<a>元素，但只发生回发(使用 JavaScript)，而不请求新页面。ImageButton 也是如此，只是显示一个图像，用户可以单击它以触发回发。



默认情况下，Button 控件使用 HTML POST 操作提交页面。LinkButton 和 ImageButton 控件则不能直接支持 HTML POST 操作。因此，使用这些按钮时，它们将客户端脚本添加到页面以允许控件以编程方式提交页面。

⦿　Image 和 ImageMap

这两个控件用于在浏览器中显示图像。ImageMap 允许在图像上定义“热点”，当单击时，要么引起一个到服务器的回发，要么导航到另一个页面。

⦿　AdRotator

这个控件允许在 Web 站点上显示随机广告。这些广告来自在服务器上创建的 XML 文件。每次刷新页面时都将更改显示的广告。广告可以加权以控制广告条的优先级别，这可以使某些广告的显示频率比其他广告高；也能编写在广告间循环的自定义逻辑。由于它缺少像单击跟踪和记录这样的大多数最简单的情况所必需的高级功能，因此这个控件在当今的 Web 站点中用得不是太多。

⦿　Calendar

Calendar 控件提供了一个功能丰富的接口，允许用户选择日期。在本章最后讨论 ASP.NET 状态引擎时将介绍它的一些用法。



⦿　FileUpload

FileUpload 控件允许用户上传可以存储在服务器上的文件。

⦿　HiddenField

HiddenField 控件可用来将数据存储在各个请求提交的页面中。如果希望页面记住特定数据，而又不希望用户在页面中看到，那么该控件就很有用。由于这个字段会显示在页面的 HTML 源代码中，因此终端用户可以访问，所以不要在其中存储任何敏感数据。

⦿　Literal、Localize 和 Substitute

这 3 个控件看起来有些像 Label 控件，因为它们都可以显示静态文本或 HTML。Literal 最大的优点是它本身不呈现额外的标记。它仅显示赋予 Text 属性的信息，因此对于显示 HTML 或者显示在 Code Behind 中构建的或从数据库检索的 JavaScript 非常有用。

Literal 控件常用的属性是 Mode 属性，该属性用于指定控件对用户所添加的标记的处理方式。可以将 Mode 属性设置为 Transform(将对添加到控件中的任何标记进行转换，以适应请求浏览器的协议)、PassThrough(添加到控件中的任何标记都将按原样呈现在浏览器中)和 Encode(使用 HtmlEncode 方法对添加到控件中的任何标记进行编码，这会将 HTML 编码转换为其文本表示形式)。

Localize 控件用在使用多种语言的 Web 站点中，并且能够从翻译后的资源文件中检索其内容。Substitute 控件用在高级缓存场景中，并且允许仅更新部分没有完全缓存的页面。

⦿　Table

<asp:Table>控件在很多方面等同于 HTML 中的<table>控件。然而，由于该控件位于服务器上，因此可以对它进行编程，动态地创建新的列和行，以及向其中添加动态数据。

◉ XML

XML 控件允许将数据从 XML 格式转换为另一种格式(如 XHTML)，以便显示在页面上。

3.2.2 HTML 控件

工具箱的 HTML 类别中包含许多 HTML 控件，它们看起来与标准类别中的控件很相似。例如，Input (Button)控件看起来就像<asp:Button>。类似地，Select 控件有<asp:DropDownList>和<asp:ListBox>作为它的对应控件。

在 VWD 2010 中，从工具箱添加到页面上的 HTML 控件只是已设置了某些属性的 HTML 元素，当然也可通过输入 HTML 标记在源视图中创建 HTML 元素。

默认情况下，ASP.NET 文件中的 HTML 元素被视为传递给浏览器的标记，作为文本进行处理，并且不能在服务器端代码中引用这些元素，只能在客户端通过 JavaScript 和 VBScript 等脚本语言来控制。若要使这些元素能以编程方式进行访问，可以通过添加 runat="server"属性表明应将 HTML 元素作为服务器控件进行处理，这样就可以使用基于服务器的代码对其进行编程引用了。

标准控件和 HTML 控件之间似乎有一些重叠，但是 HTML 控件的功能比标准类别中的控件的功能少得多。一般来说，标准类别中的真正服务器控件提供了更多的功能，无论是在 VWD 中的设计时支持方面还是在运行时能做的事情方面都是如此。不过这种功能是有代价的。因为它们增加了复杂度，所以处理服务器控件会多花一点时间。然而，在大多数 Web 站点上，用户可能不会注意到这一差别。只有当有一个高通信量的 Web 站点且在页面上有很多控件时，使用 HTML 控件才会提供稍好一些的性能。

在大多数情况下，人们更愿意使用服务器控件而不是与它们对应的 HTML 控件。因为服务器控件提供了更多的功能，在页面中更灵活，可以给用户带来更丰富的体验，而且有比较好设计时支持，因此值得选择。如果十分确信不需要服务器控件提供的这些功能，则可以选择 HTML 控件。

3.2.3 数据控件

数据控件是在 ASP.NET 2.0 中引入的，它提供了非常方便的方式来访问各种数据源，如数据库、XML 文件以及对象。使用数据控件，不需要像在 ASP.NET 的早期版本中那样编写很多的代码才能访问数据源，而只需把数据控件指向适当的数据源即可，ASP.NET 运行库会负责处理大部分难题。在本书的第 5 章中将重点介绍这些控件的使用。

3.2.4 验证控件

ASP.NET 4.0 为开发人员提供了一套完整的服务器控件来验证用户输入的信息是否有效，这些控件可与 ASP.NET 网页上的任何控件(包括 HTML 和服务器控件)一起使用。



有效性验证控件最出色的是它们能在客户端和服务器上检查输入。当向 Web 页面中添加一个有效性验证控件时，控件就会呈现在客户端验证关联控件有效性的 JavaScript。大多数启用了 JavaScript 的现代 Web 浏览器(包括 IE、Firefox、Chrome、Opera 和 Safari)都能进行这种客户端有效性验证。同时，有效性验证也可以在服务器上自动进行。这样就容易向用户提供关于使用客户端脚本的数据的即时反馈，从而使 Web 页面在服务器上免受伪数据的侵扰。

1. 验证控件简介

ASP.NET 提供了 6 个有效性验证控件，其中 5 个控件用来执行实际的有效性验证，而只有 ValidationSummary 控件用来向用户呈现页面中出现的错误的反馈信息。如表 3-4 所示列出了 ASP.NET 提供的验证控件及其功能说明。

表 3-4　ASP.NET 验证控件

验 证 类 型	使用的控件	说　明
必选项	RequiredFieldValidator	验证一个必填字段，如果该字段没填，那么将不能提交信息
与某值的比较	CompareValidator	将用户的输入与一个常数值或者另一个控件或特定数据类型的值进行比较(使用小于、等于或大于等比较运算符)，同时也可以用来校验控件中内容的数据类型，例如整形、字符串型等。典型的例子有验证密码和确认密码两个字段是否相等
范围检查	RangeValidator	RangeValidator 控件可以用来判断用户输入的值是否在某一特定范围内。可以检查数字对、字母对和日期对限定的范围。属性 MaximumValue 和 MinimumValue 用来设定范围的最大值和最小值
模式匹配	RegularExpressionValidator	它根据正则表达式来验证用户输入字段的格式是否合法，如电子邮件、身份证、电话号码等。ControlToValidate 属性确定需要验证的控件，ValidationExpression 属性则确定需要验证的表达式的样式
用户定义	CustomValidator	使用自己编写的验证逻辑检查用户输入。此类验证能够检查在运行时派生的值。在运行定制的客户端 JavaScript 或 VBScript 函数时，可以使用这个控件
验证汇总	ValidationSummary	该控件不执行验证，但该控件将本页所有验证控件的验证错误信息汇总为一个列表并集中显示，列表的显示方式由 DisplayMode 属性设置



表 3-4 的前 5 个验证控件基本上都继承自同一个基类，因此它们有一些共同的行为，5 个有效性验证控件中的 4 个以相同的方式操作，并包含允许验证关联控件的内置行为，CustomValidator 控件则允许用户写非内置的自定义功能。如表 3-5 所示为有效性验证控件共有的常用属性。

表 3-5　有效性验证控件的共有属性

属　性	说　明
Display	这个属性确定隐藏的错误消息是否占用空间。如果将 Display 设置为 Static，错误消息就会占用屏幕空间，即使在隐藏时也是如此；如果设置为 None，错误消息就看不到
CssClass	这个属性允许设置应用到错误消息文本的 CssClass 特性
ErrorMessage	这个属性保存用在 ValidationSummary 控件中的错误消息。当 Text 属性为空时，也用 ErrorMessage 值作为出现在页面上的文本
Text	Text 属性用作有效性验证控件显示在页面上的文本。它可以是一个星号(*)以表示出现一个错误，也可以是具体的文本信息
ControlToValidate	这个属性包含需要验证有效性的控件的 ID
EnableClientScript	这个属性用于确定控件是否提供客户端的有效性验证，默认为 True
SetFocusOnError	这个属性确定客户端脚本是否将焦点放在产生错误的第一个控件上，默认值为 False
ValidationGroup	有效性验证控件可以组合在一起，允许对选中的控件进行有效性验证。同一个 ValidationGroup 中的所有控件都会被同时检查，这意味着如果控件不是这个控件组的一部分，就不对它进行有效性验证
IsValid	通常在设计时不会设置这个属性，不过在运行时它提供关于是否通过了有效性验证测试的信息


知识点

乍一看，Text 和 ErrorMessage 属性的作用似乎是一样的。它们都可以用来以错误消息的形式向用户提供反馈。但是当与 ValidationSummary 控件结合起来使用时，两者之间就有了细微的区别。当同时设置这两个属性时，Validation 控件显示 Text 属性，而 ValidationSummary 控件则显示 ErrorMesage。

除了上述共有属性之外，RangeValidator 控件还有如表 3-6 所示的重要属性。

表 3-6　RangeValidator 控件的重要属性

属　性	说　明
MinimumValue	该属性确定可接受的最小值。例如，当检查 1～10 之间的整数时，将该属性设置为 1
MaximumValue	该属性确定可接受的最大值。例如，当检查 1～10 之间的整数时，将该属性设置为 10
Type	该属性确定有效性验证控件检查的数据类型，可以设置为 String、Integer、Double、Date 或 Currency 来检查各自的数据类型

CompareValidator 控件能用来比较一个控件的值与另一个控件的值。它通常用在注册表单中，用户必须输入两次密码，以确保两次输入的密码相同。也可以不与另一个控件作比较，而是与一个常量值比较。CompareValidator 控件的其他属性如表 3-7 所示。

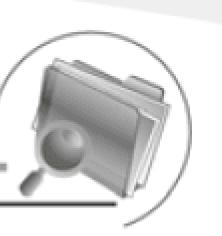


表 3-7　CompareValidator 控件的其他属性

属　　性	说　　明
ControlToCompare	该属性包含验证器要与之比较的控件 ID。设置了该属性，ValueToCompare 就无效了
Operator	该属性确定比较操作的类型。如，当 Operator 设置为 Equal 时，两个控件都必须包含验证器认为有效的同一个值。类似地，还有一些选项，如 NotEqual、GreaterThan 和 GreaterThanEqual，用来执行不同的有效性验证操作
Type	该属性确定有效性验证控件检查的数据类型，可以设置为 String、Integer、Double、Date 或 Currency 来检查各自的数据类型
ValueToCompare	该属性允许定义一个要比较的常量值。它通常用在必须输入 Yes 这样的单词的协议中，表示同意某些条件。只要将 ValueToCompare 设置为单词 Yes，并将 ControlToValidate 设置为要验证有效性的控件即可



ValidationSummary 控件向用户提供了它从单个有效性验证控件的 ErrorMessage 属性中检索到的一个错误列表。它能以 3 种不同的方式显示这些错误：使用一个嵌在页面中的列表，使用 JavaScript 警报框或者同时使用这两种方式。可以通过 ShowMessageBox 和 ShowSummary 属性控制这个设置。此外，通过 DisplayMode 属性可以修改表现错误列表的方式，默认设置为 BulletList，其中每个错误都是项目列表中的一个项。

2. 使用验证控件

【例 3-4】演示验证控件的使用。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-4】，单击【确定】按钮。

(2) 打开 Default.aspx 文件的设计视图，选择【表】|【插入表】命令，打开【插入表格】对话框，插入一个 9 行 3 列的表格，如图 3-20 所示。

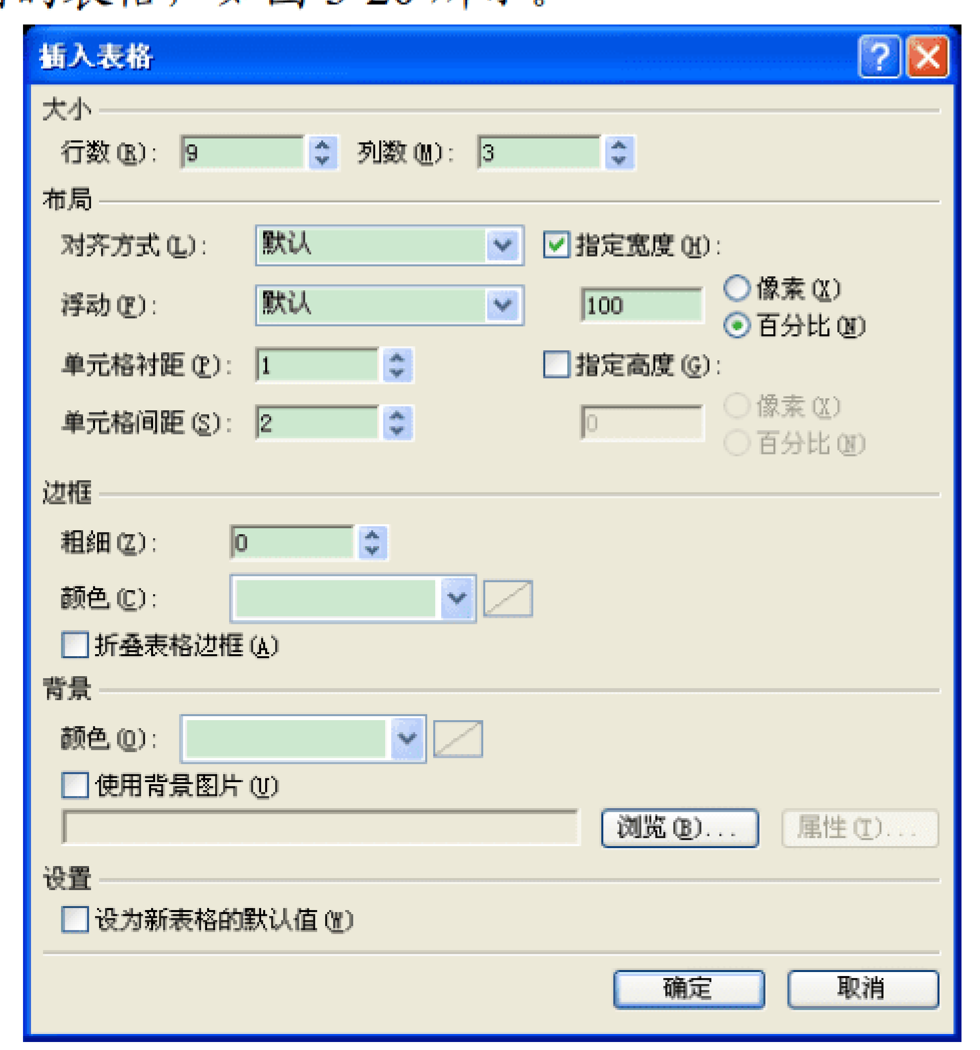


图 3-20　【插入表格】对话框

(3) 合并第一行的 3 个单元格。选择这 3 个单元格，选择【表】|【修改】|【合并单元格】命令。

(4) 在第一行的单元格中输入“请输入以下信息”。在第 2~8 行的单元格中，第一列单元格中输入文本信息，第二列单元格中添加用于输入信息的 TextBox 控件，第三列单元格中添加相应的验证控件。在最后一行中，第一列添加一个 Button 控件，用于提交表单数据；第二列添加一个 Label 控件，用于显示提示信息；第三列添加一个 ValidationSummary 控件，显示验证错误信息。如图 3-21 所示。

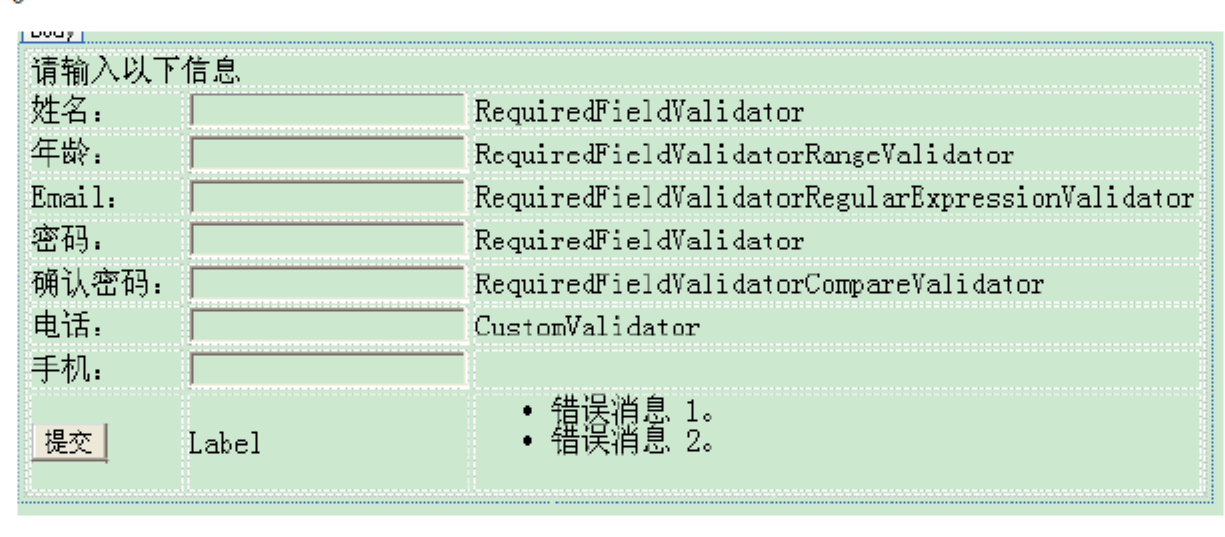


图 3-21　在单元格中添加文本信息和控件



(5) 设置 TextBox4 和 TextBox5 的 TextMode 属性为“Password”。

(6) 同时选中 5 个 RequireFieldValidator 控件，设置其 Text 属性为“*”，并且分别设置它们的 ErrorMessage 属性为“姓名不能为空”、“年龄不能为空”、“Email 不能为空”、“密码不能为空”和“确认密码不能为空”。设置每个 RequireFieldValidator 控件的 ControlToValidator 属性为其所在行的 TextBox 控件的 ID。

(7) 设置 RangeValidation 控件的 ControlToValidator 属性为“TextBox2”，Minimum Value 属性为“1”，Maximum 属性为“100”，Type 属性为“Integer”，ErrorMessage 属性为“年龄范围必须是 1~100”。

(8) 设置 RegularExpressionValidation 控件的 ControlToValidator 属性为“TextBox3”，ErrorMessage 属性为“Email 格式错误”；然后单击 ValidationExpression 属性右边的浏览按钮，在弹出的【正则表达式编辑器】对话框中选择【Internet 电子邮件地址】选项，如图 3-22 所示。

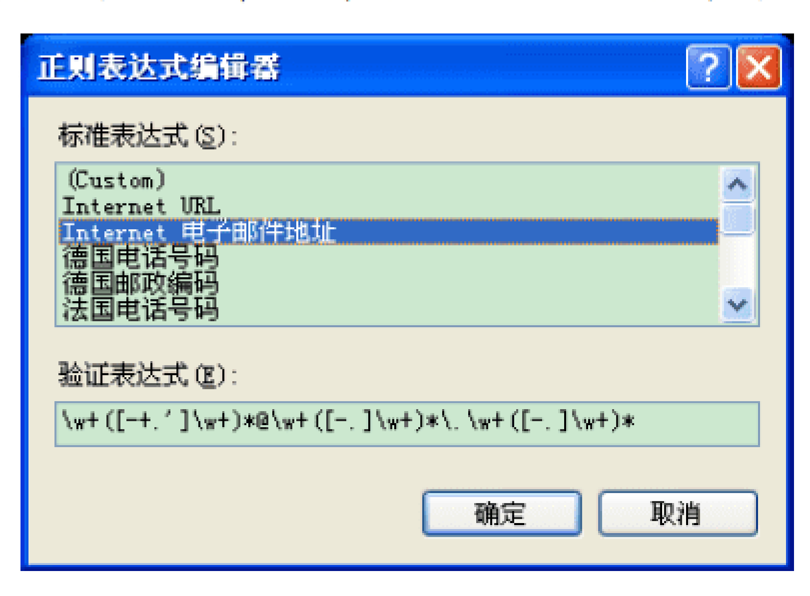


图 3-22　【正则表达式编辑器】对话框

(9) 设置 CompareValidation 控件的 ControlToValidator 属性为“TextBox5”，ControlToCompare 属性为“TextBox4”，ErrorMessage 属性为“两次输入的密码不相同，请重新输入”。



(10) 设置 CustomValidator 控件的 Display 属性为“Dynamic”，ErrorMessage 属性为“电话和手机至少要输入一个”，ClientValidationFunction 属性为“ValidatePhone”，然后为该控件添加 ServerValidate 事件处理程序。代码如下：

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (!string.IsNullOrEmpty(TextBox6.Text) ||!string.IsNullOrEmpty(TextBox7.Text))
    {
        args.IsValid = true;
    }
    else
    {
        args.IsValid = false;
    }
}
```



(11) 切换到页面的源视图，在<body>之前添加如下 JavaScript 代码：

```
<script type="text/javascript">
    function ValidatePhone(source, args) {
        var telephone = document.getElementById('<%= TextBox6.ClientID %>');
        var mobile = document.getElementById('<%= TextBox7.ClientID %>');
        if (telephone.value != '' || mobile.value != '') {
            args.IsValid = true;
        }
        else {
            args.IsValid = false;
        }
    }
</script>
```

上述 JavaScript 函数 ValidatePhone 确保在将页面提交回服务器之前至少输入了一个电话号码。

(12) 设置按钮控件的 Text 属性为“提交”，Label 控件的 Text 属性为空。为按钮控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "全部验证通过！";
}
```

(13) 编译并运行程序，在浏览器中加载页面 Default.aspx，如果没有输入任何信息，或者输入的信息不合法，系统将给出错误提示，如图 3-23 所示。

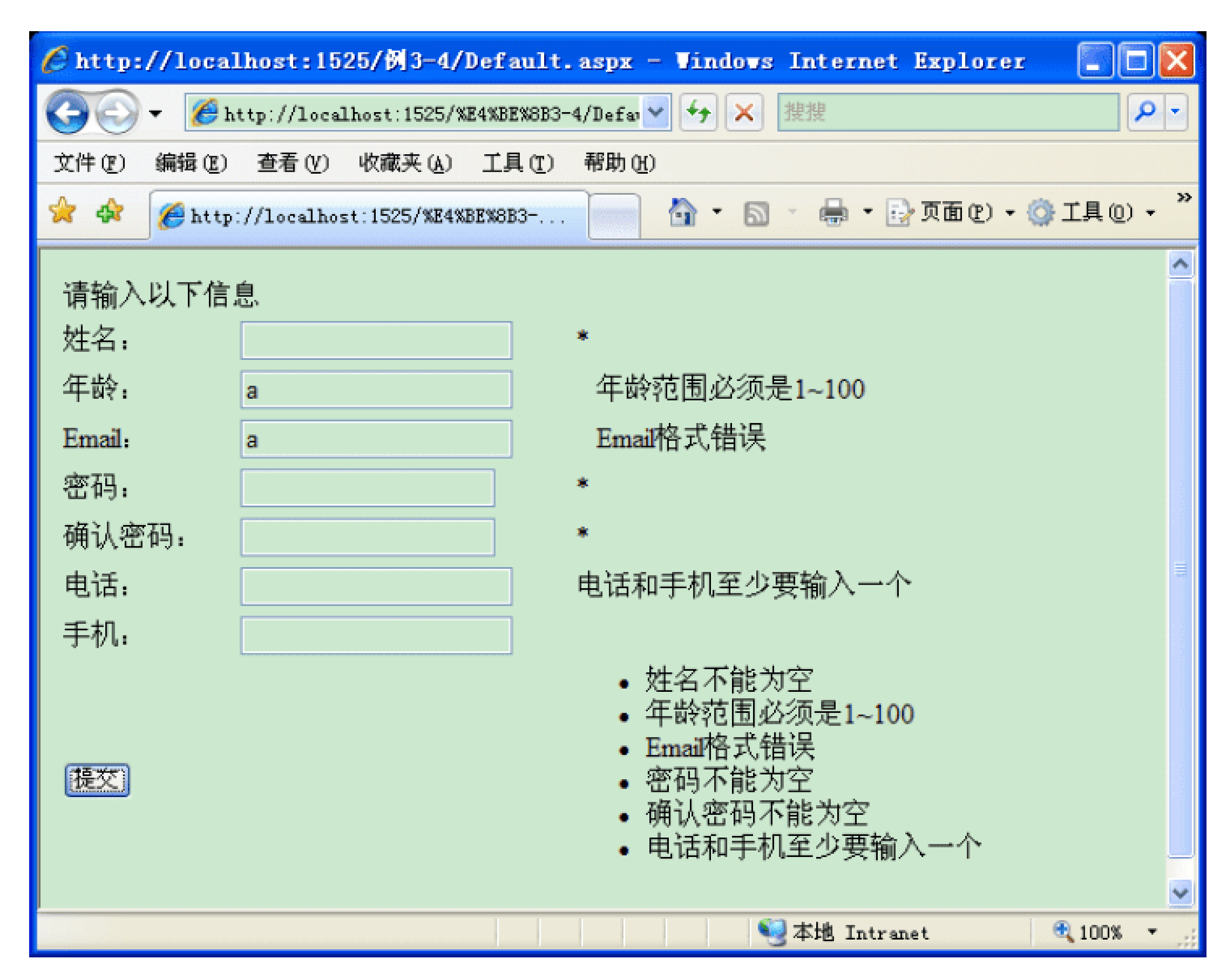


图 3-23　页面验证效果

(14) 回到 VWD 中，将 ValidationSummary 控件的 ShowMessageBox 设置为 True，ShowSummary 设置为 False，同时将它的 HeaderText 属性设置为“错误提示”。

(15) 再次编译并运行程序，在浏览器中打开页面，注意此时得到的不是含有错误的内联列表，而是一个警告对话框，如图 3-24 所示。

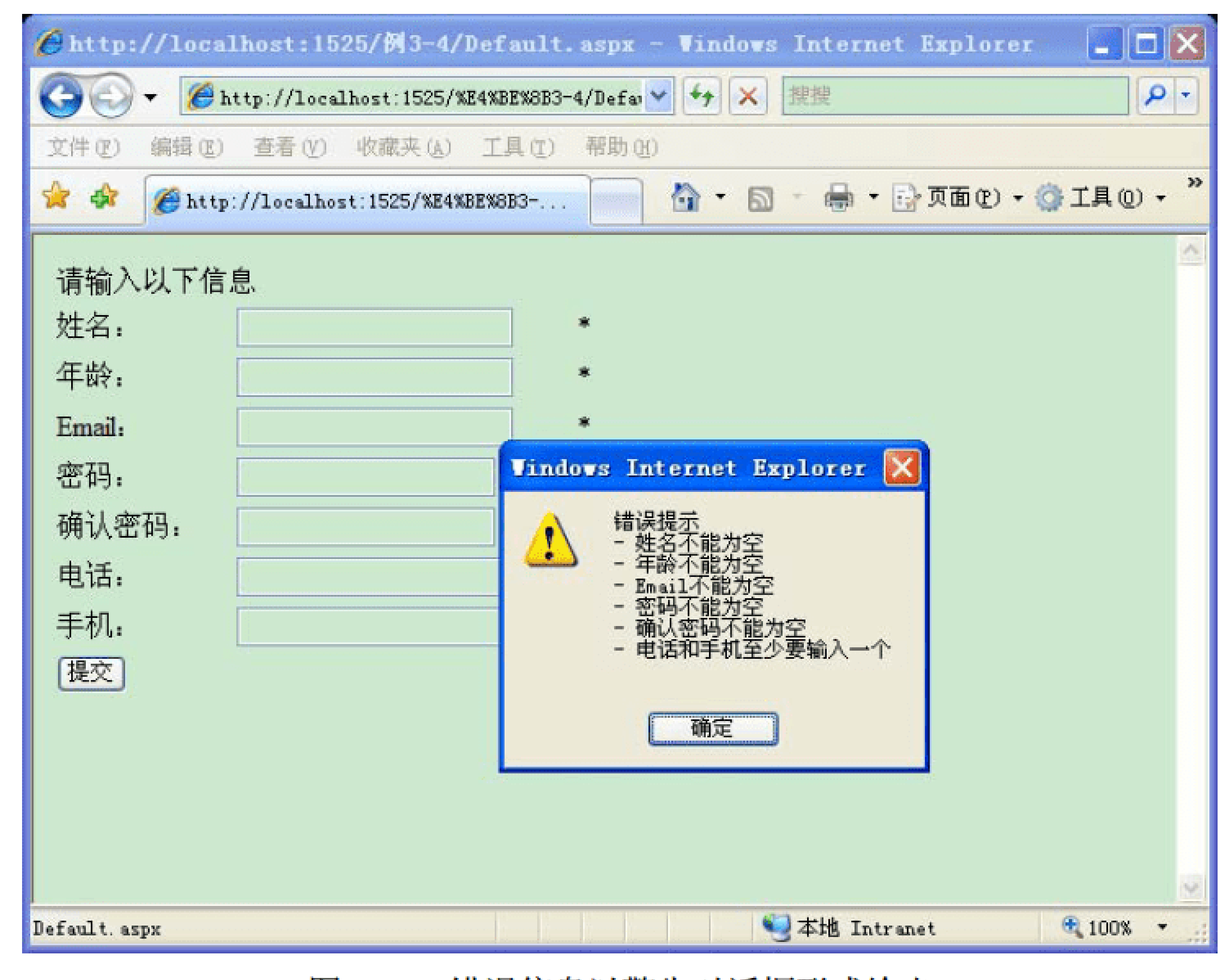
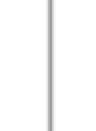


图 3-24　错误信息以警告对话框形式给出





(16) 如果信息输入全部正确，则所有验证通过，如图 3-25 所示。

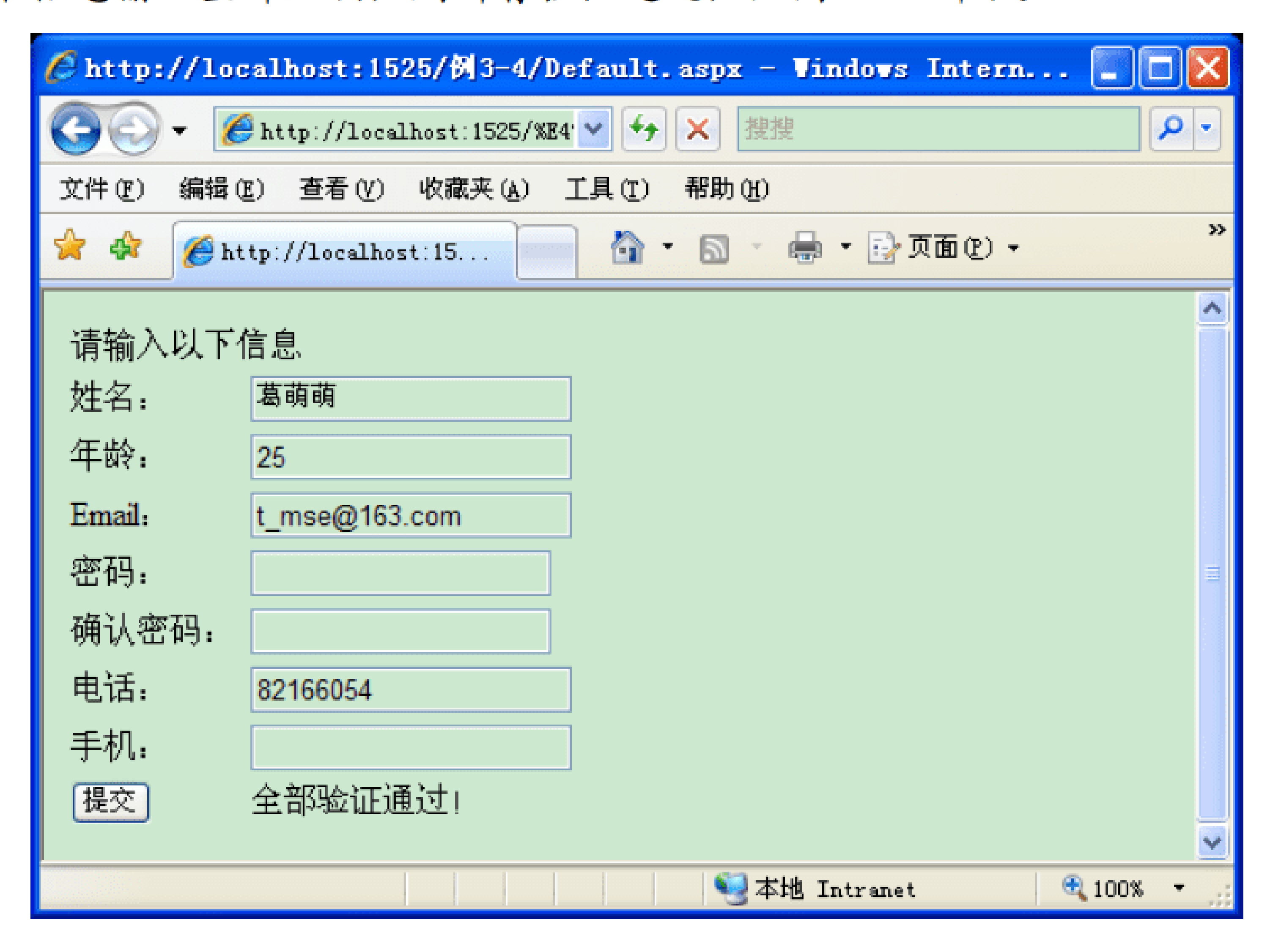


图 3-25　所有验证通过





提示

每当将一个 Web 站点发布到 Internet 上后，就失去了控制其用户的能力。为了防止恶意用户向系统中输入伪数据，总是要使用 ASP.NET 的有效性验证控件验证用户输入的有效性。

3.2.5　导航控件

当站点包含的页面比较多时，有一个稳固而清晰的导航结构就很重要，这样才能让用户顺畅地浏览站点。使用良好的导航系统，项目中所有没有连接的 Web 页面就会形成一个完整而连贯的 Web 站点。

ASP.NET 4.0 提供了 3 个有用的导航控件：SiteMapPath、Menu 和 TreeView。

- SiteMapPath：这个 Web 控件提供一个面包条(breadcrumb)，它是一行文本，显示用户当前在网站结构中的位置。例如，在网上书店中，如果用户浏览到《Visual C++》时，面包条可能类似于“主页->计算机->编程类->Visual C++”，其中每部分(如主页，计算机等)都显示为返回到前一部分的链接。面包条能够让用户快速地查看当前在网站中的位置，并沿逻辑层次结构向上导航。
- Menu：这个 Web 控件提供网站结构的层次视图。对于学校的网站，顶层菜单将包含主类别(如学校介绍、机构设置、新闻等)，每个菜单项又可以包含各自的子菜单，显示各自的子类别。

- TreeView：树视图提供了与菜单相同的数据，唯一的区别是显示数据的方式。树视图显示为可展开或可折叠的树，而菜单(Menu)是由菜单项和子菜单组成。

一般情况下，开发人员利用站点地图和 SiteMapPath 控件实现自动导航，利用 Menu 控件或者 TreeView 控件实现自定义导航。

1. 创建站点地图

为了更容易地使用 Menu、TreeView 或 SiteMapPath 显示站点中的相关页面，ASP.NET 使用一个基于 XML 的文件来描述 Web 站点的逻辑结构。默认情况下，这个文件名为 Web.sitemap。然后站点中的导航控件会用这个文件以有组织的方式表现相关的链接。只要将一个导航控件与这个 Web.sitemap 文件挂勾，就能创建出复杂的用户界面元素，如折叠菜单或树型视图等。



知识点

默认情况下，应将站点地图文件命名为 Web.sitemap，这样控件就可以自动找到正确的文件。对于更高级的情况，可以有多个不同名称的站点地图文件，且在向系统提供这些附加文件的 Web.config 中有一个配置设置。在大多数情况下，有一个站点地图文件就足够了。

站点地图文件的基础版本如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
   <siteMapNode url="~/" title="Home" description="Go to the homepage">
      <siteMapNode url="~/Reviews" title="Reviews" description="Reviews published on this site" />
      <siteMapNode url="~/About" title="About"   description="About this site" />
   </siteMapNode>
</siteMap>
```

这个站点地图文件的根节点是 siteMap，其下使用 siteMapNode 节点建立层次结构，每个 siteMapNode 可以有多个子节点(但是，在 siteMap 元素下只能有一个 siteMapNode)，子节点仍然用<siteMapNode>定义，可以用来创建一个既有广度又有深度的站点结构。在本例中，只有一个名为 Home 的根节点，它包含两个子元素：Reviews 和 About。本例中的 siteMapNode 元素有 3 个属性：url、title 和 description。

- url 属性应指向 Web 站点中的有效页面。可以用“~”语法来引用基于应用程序根文件夹的 URL。
- title 属性用作显示页面的名称。在使用 Menu、TreeView 和 SiteMapPath 控件时将看到关于它的更多信息。
- description 属性用作导航元素的工具提示。





提示

虽然 ASP.NET 运行库不允许多次指定同一个 URL，但是可以通过添加一个查询字符串使 URL 唯一以绕过这一问题。例如，~/Login.aspx 和~/Login.aspx?type=Admin 会被看作是两个不同的页面。

为了能够使用 Web.sitemap 文件，ASP.NET 使用了 SiteMapDataSource 控件，这是一个数据控件。当使用 SiteMapPath 控件显示“痕迹导航”时，ASP.NET 会自动找到 Web.sitemap 文件。使用另外两个导航控件时，则需要显式地指定一个 SiteMapDataSource 作为 Web.sitemap 文件的中间层。

VWD 没有自动基于当前站点的结构创建站点地图文件的方式。要创建一个有用的 Web.sitemap 文件，需要向站点添加一个文件，然后手动向该文件添加必需的 siteMapNode 元素。

【例 3-5】创建 Web.sitemap 文件。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-5】，单击【确定】按钮。



(2) 在【解决方案资源管理器】面板中，右击【例 3-5】解决方案，从弹出的快捷菜单中选择【添加新项】命令，在打开的【添加新项】对话框中选择【站点地图】选项，保持默认名称为 Web.sitemap，单击【添加】按钮。

(3) 新添加的 Web.sitemap 文件中会出现一个包含两个子节点的根元素，修改 Web.sitemap 文件如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
<siteMapNode url="~/Default.aspx" title="首页" description="首页">
  <siteMapNode url="~/Info.aspx" title="学校简介" description="学校简介" >
  </siteMapNode>
  <siteMapNode url="~/Depart.aspx" title="院系设置" description="院系设置" >
    <siteMapNode url="~/Depart1.aspx" title="软件学院" description="软件学院" />
    <siteMapNode url="~/Depart2.aspx" title="管理学院" description="管理学院" />
  </siteMapNode>
</siteMapNode>
</siteMap>
```

(4) 保存文件，完成站点地图的创建。

Web.sitemap 文件本身用处并不大，需要向站点中添加导航控件来使用站点地图。下面将介绍如何使用导航控件来实现网站导航。

添加站点地图到应用程序中时，需要将站点地图放在 Web 应用程序的根目录下，并保持其文件名为 Web.sitemap。如果将该文件放在另一个文件夹中或使用不同的文件名， SiteMapPath 导航控件将不能找到站点地图，从而不能知道网站的结构，因为默认情况下 SiteMapPath 导航控

件在根目录下寻找名为 Web.sitemap 的文件。


提示

内部没有内容的 XML 元素可以采用两种形式的结束标签：一种是冗余方式，如<myTag attribute="value"...></myTag>，另一种是使用简洁方式，如<myTag attribute="value".../>。

2. 使用 SiteMapPath 控件

定义好站点地图之后，就可以使用 SiteMapPath 控件显示导航路径了，也就是显示当前页面在网站中的位置。只需要将该控件拖放到站点地图中包含的.aspx 页面上，它就会自动实现导航，不需要开发者编写任何代码。

SiteMapPath 控件显示了当前在站点结构中的位置。它将自身表现为一系列链接，常称之为痕迹导航(breadcrumb)。它是一个非常简单但功能强大的控件，有 50 多个公共属性，可以通过【属性】面板来设置这些属性。

提示

只有包含在站点地图中的网页才能被 SiteMapPath 控件导航；如果将 SiteMapPath 控件放置在站点地图中未列出的网页中，该控件将不会显示任何信息。

SiteMapPath 控件像大多数 Web 控件一样，也有很多可用于定制其外观的属性。如表 3-8 所示为 SiteMapPath 控件的常用属性。

表 3-8　SiteMapPath 控件的常用属性

属　性	说　明
CurrentNodeStyle	定义当前节点的样式，包括字体、颜色、样式等
NodeStyle	定义导航路径上所有节点的样式
ParentLevelsDisplayed	指定在导航路径上显示的相对于当前节点的父节点层数。默认值为-1，表示父级别数没有限制
PathDirection	指定导航路径上各节点的显示顺序，默认值为 RootToCurrent，即按从左到右的顺序显示从根节点到当前节点的路径；另一选项为 CurrentToRoot，即按相反的顺序显示导航路径
PathSeparator	指定导航路径中节点之间的分隔符。默认值为>，也可自定义为其他符号
PathSeparatorStyle	定义分隔符的样式
RenderCurrentNodeAsLink	是否将导航路径上当前页名称显示为超链接，默认值为 false
RootNodeStyle	定义根节点的样式
ShowToolTips	当鼠标悬停于导航路径的某个节点时，是否显示相应的工具提示信息，默认值为 true，即当鼠标悬停于某节点上时，显示该节点在站点地图中定义的 Description 属性值



下面通过具体例子演示如何利用前面介绍的站点地图和 SiteMapPath 控件来实现自动导航。

【例 3-6】在例【3-5】的基础上，利用 SiteMapPath 控件实现自动导航。

(1) 启动 VWD 2010，打开网站【例 3-5】。

(2) 在【解决方案资源管理器】中，分别添加名为 Info.aspx、Depart.aspx、Depart1.aspx、Depart2.aspx 的网页。

(3) 在每个页面中都添加一个 SiteMapPath 控件，在设计视图中可以看到该页面的导航路径，如图 3-26 所示为 Depart2.aspx 的设计视图效果。可见，利用站点地图和 SiteMapPath 控件实现自动导航非常方便。

首页 > 院系设置 > 管理学院

图 3-26　添加 SiteMapPath 控件后的效果

(4) 编译并运行程序，在默认浏览器中打开不同的页面，体验导航的便捷。

如果要在客户端查看源文件，可以看到，SiteMapPath 呈现为一系列包含一个链接或纯文本的<span>元素。

3. 使用 Menu 控件

Menu 控件主要用于创建一个菜单，让用户快速选择不同的页面，从而完成导航功能。该控件可以包含一个主菜单和多个子菜单。菜单有动态和静态两种显示模式。静态显示模式是指定义的菜单始终完全显示，动态显示模式是指需要用户将鼠标停留在菜单项上时才显示子菜单。

Menu 控件的常用属性如表 3-9 所示。

表 3-9　Menu 控件的常用属性

属　性	说　明
DynamicEnableDefaultPopOutImage StaticEnableDefaultPopOutImage	是否在菜单各项之间显示分隔图像，默认值为 true
DynamicPopOutImageUrl StaticPopOutImageUrl	设置菜单中自定义分隔图像的 URL
DynamicBottomSeparatorImageUrl StaticBottomSeparatorImageUrl	指定在菜单项下方显示分隔图像的 URL，默认值为空字符串("")，即菜单项下方不显示任何图像
DynamicTopSeparatorImageUrl StaticTopSeparatorImageUrl	指定在菜单项上方显示分隔图像的 URL，默认值为空字符串("")，即菜单项上方不显示任何图像
DynamicHorizontalOffset StaticHorizontalOffset	指定菜单相对于其父菜单的水平距离，单位是像素，默认值为 0；该属性值可正可负，为负值时，各菜单之间的距离会缩小
DynamicVerticalOffset StaticVerticalOffset	指定菜单项相对于其父菜单项的垂直距离
MaximumDynamicDisplayLevels	设置动态菜单的最大层数，默认值为 3
Orientation	设置菜单的展开方向，有 Horizontal 和 Vertical 两个选项，默认值为 Vertical，即垂直方向



(续表)

属　　性	说　　明
Items	获取包含 Menu 控件中的所有菜单项，返回 MenuItemCollection 对象
DataSourceID	SiteMapDataSource 控件的 ID，为 Web.sitemap 文件中的菜单提供数据
RenderingMode	ASP.NET 4.0 中新增的属性。这个属性用于确定控件是使用表和内联样式，还是使用无序列表和 CSS 样式来显示自身
IncludeStyleBlock	ASP.NET 4.0 中新增的属性。这个属性使得开发人员可以完全控制控件的样式

Menu 控件的用法非常灵活，设计者可以利用它定义各种菜单样式，实现类似于 Windows 窗口菜单的功能。

下面通过一个具体的例子演示如何利用 Menu 控件实现自定义导航。

【例 3-7】使用 Menu 控件在网页中添加一个菜单，实现自定义导航功能。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-7】。

(2) 在应用程序中分别添加名为 One.aspx、Two1.aspx、Two2.aspx、Two3.aspx、Three1.aspx 和 Three2.aspx 的网页。

(3) 打开 Default.aspx 文件的设计视图，向页面拖放一个 Menu 控件。

(4) 将 Menu 控件的 Orientation 属性设置为 Horizontal，以便使其横向排列。

(5) 单击 Menu 控件右上方的小三角，打开【Menu 任务】面板，选择【编辑菜单项】命令，如图 3-27 所示。将打开【菜单项编辑器】对话框，如图 3-28 所示。在此输入各级菜单项。

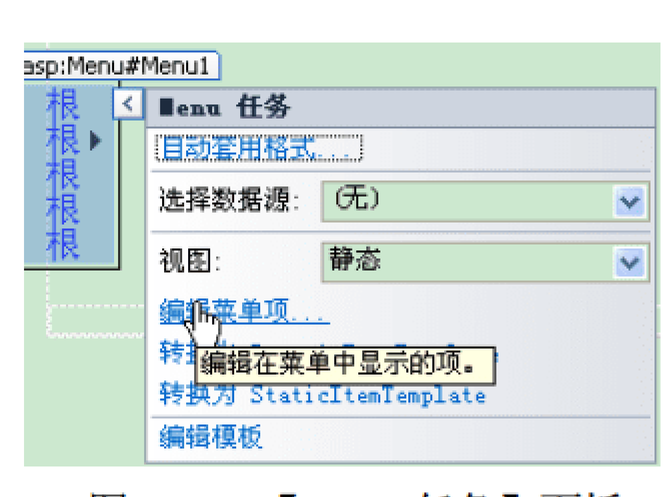


图 3-27　【Menu 任务】面板

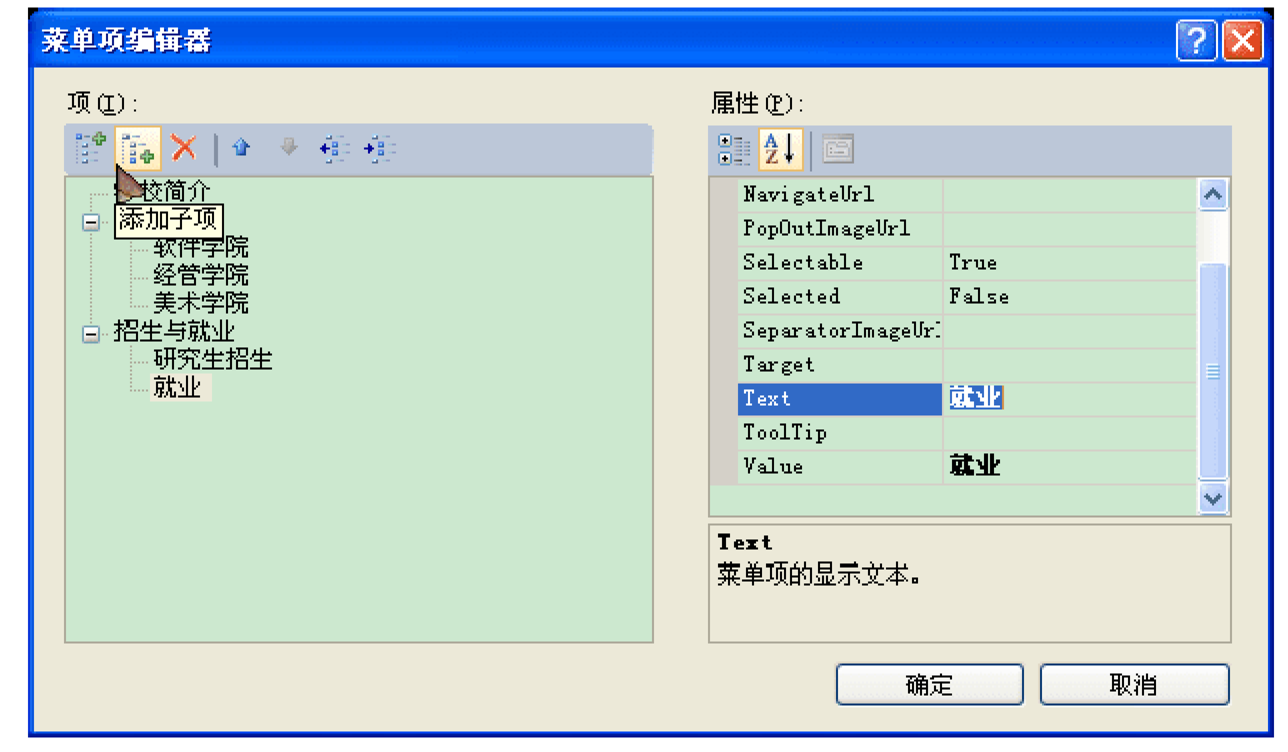
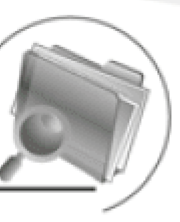


图 3-28　【菜单项编辑器】对话框

(6) 在【菜单项编辑器】窗口右侧的属性选项中，利用 NavigateUrl 属性设置各菜单项链接的网页，可以单击该属性右侧的浏览按钮，打开【选择 URL】对话框，选择当前网站内的页面，如图 3-29 所示。全部设置完成后，单击【确定】按钮。



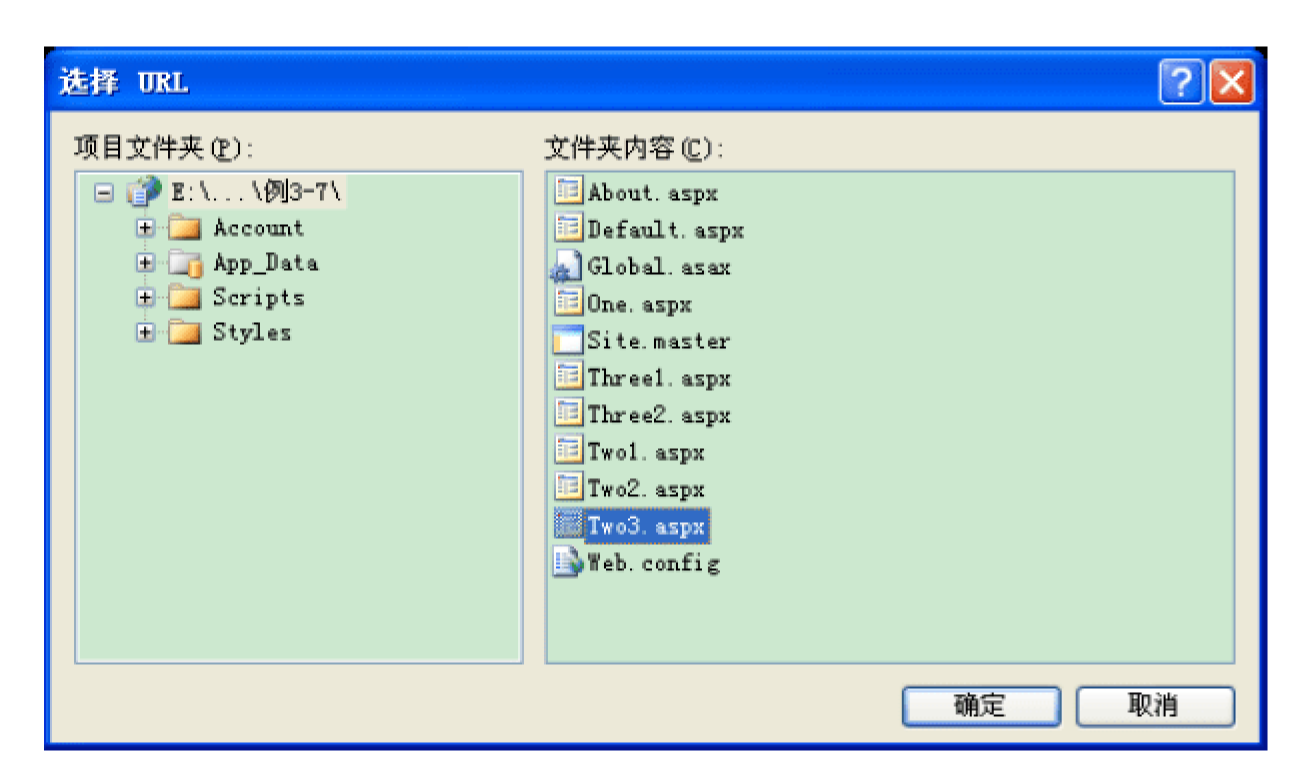


图 3-29　【选择 URL】对话框

(7) 为了使导航菜单更美观，可以给 Menu 控件设置一下格式，单击 Menu 控件右上方的小三角，打开【Menu 任务】面板，选择【自动套用格式】命令，打开【自动套用格式】对话框，选择【彩色型】架构，单击【确定】按钮，如图 3-30 所示。

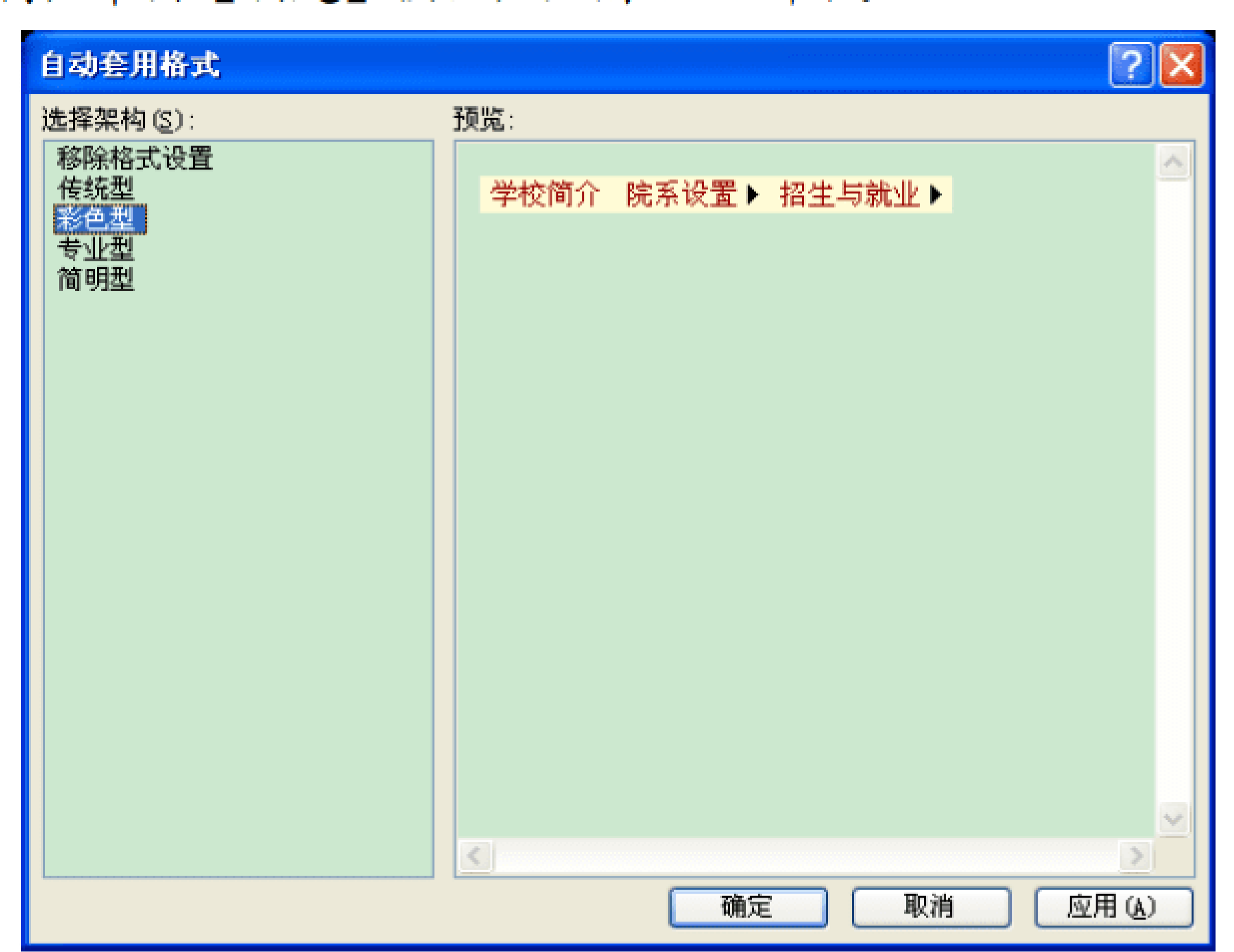


图 3-30　【自动套用格式】对话框

(8) 切换到 Default.aspx 的源视图，可以看到生成的 Menu 控件的代码如下:

```
<asp:Menu ID="Menu1" runat="server" BackColor="#FFFBD6"
    DynamicHorizontalOffset="2" Font-Names="Verdana" Font-Size="0.8em"
    ForeColor="#990000" Orientation="Horizontal" StaticSubMenuIndent="10px">
    <DynamicHoverStyle BackColor="#990000" ForeColor="White" />
    <DynamicMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
    <DynamicMenuStyle BackColor="#FFFBD6" />
    <DynamicSelectedStyle BackColor="#FFCC66" />
    <Items>
        <asp:MenuItem NavigateUrl="~/One.aspx" Text="学校简介" Value="学校简介"></asp:MenuItem>
```





```
            <asp:MenuItem Text="院系设置" Value="院系设置">
                <asp:MenuItem NavigateUrl="~/Two1.aspx" Text="软件学院" Value="软件学院">
                </asp:MenuItem>
                <asp:MenuItem NavigateUrl="~/Two2.aspx" Text="经管学院" Value="经管学院">
                </asp:MenuItem>
                <asp:MenuItem NavigateUrl="~/Two3.aspx" Text="美术学院" Value="美术学院">
                </asp:MenuItem>
            </asp:MenuItem>
            <asp:MenuItem Text="招生与就业" Value="招生与就业">
                <asp:MenuItem NavigateUrl="~/Three1.aspx" Text="研究生招生" Value="研究生招生">
                </asp:MenuItem>
                <asp:MenuItem NavigateUrl="~/Three2.aspx" Text="就业" Value="就业"></asp:MenuItem>
            </asp:MenuItem>
        </Items>
        <StaticHoverStyle BackColor="#990000" ForeColor="White" />
        <StaticMenuItemStyle HorizontalPadding="5px" VerticalPadding="2px" />
        <StaticSelectedStyle BackColor="#FFCC66" />
    </asp:Menu>
```

(9) 编译并运行程序，在浏览器中打开 Default.aspx 网页，效果如图 3-31 所示。

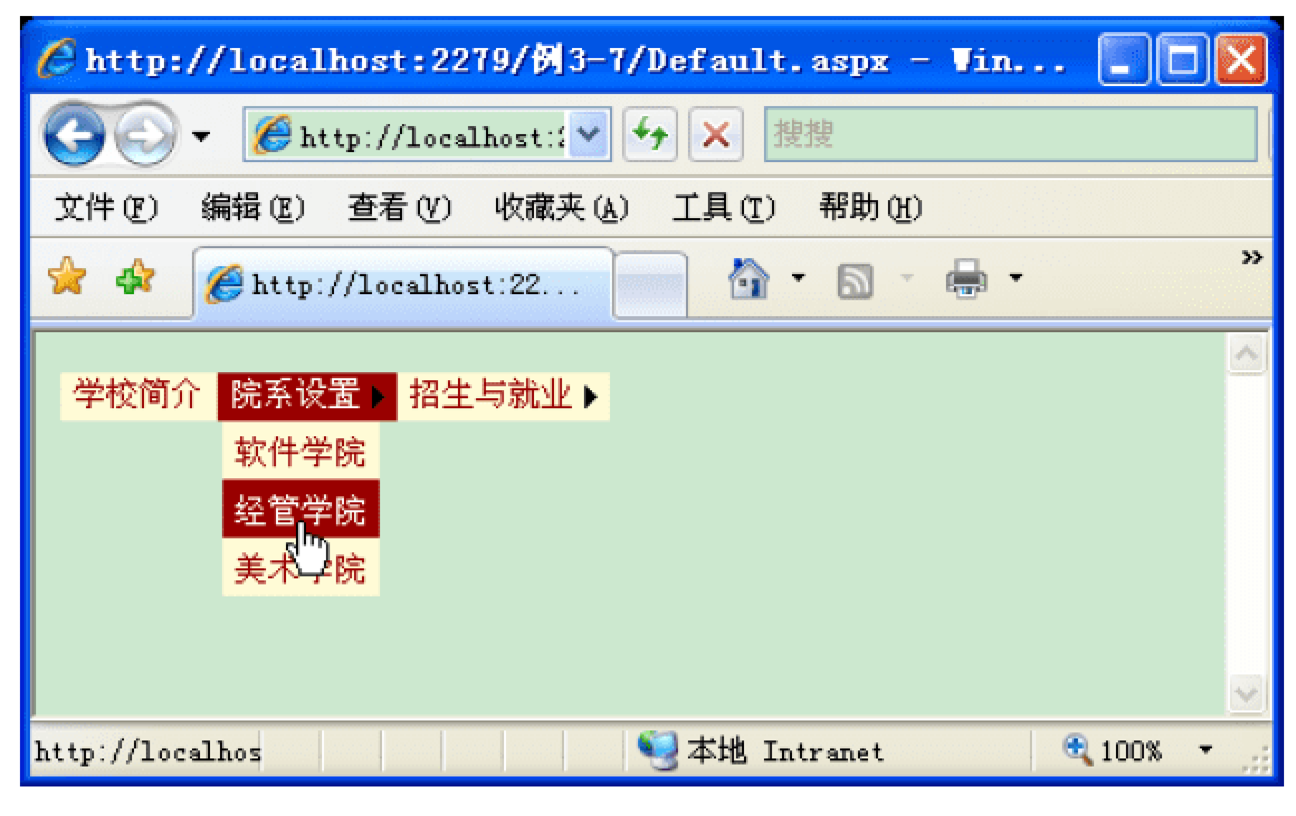
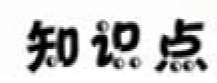


图 3-31　Menu 控件实现的导航效果



知识点

除了可以通过菜单项编辑器编辑 Menu 控件的菜单项之外，还可以为其指定一个数据源，通常该数据源是 SiteMapDataSource 控件。有关该控件的使用将在介绍 TreeView 控件时一起介绍。

4. 使用 TreeView 控件

TreeView 控件与 Menu 控件相似，都提供了导航功能。TreeView 控件与 Menu 控件的区别





是它不再像 Menu 控件由菜单项和子菜单组成，而是用一个可折叠树显示网站的各个部分。其根节点下可以包含多个子节点，子节点下又可以包含子节点，最下层是叶节点。访问者可以快速看到网站的所有部分及位于网站结构层次中的位置。树中的每个节点都显示为一个超链接，被单击时把用户引导到相应的部分。

TreeView 控件也包含很多属性，其中常用属性如表 3-10 所示。

表 3-10　TreeView 控件的常用属性

属　　性	说　　明
CollapseImageUrl	节点折叠后显示的图像。默认情况下，常用带方框的+号作为可展开指示图像
CollapseImageToolTip	当用户将鼠标悬停在可折叠菜单项上时显示的工具提示
ExpandImageUrl	节点展开后显示的图像。默认情况下，常用带方框的-号作为可折叠指示图像
EnableClientScript	是否可以在客户端处理节点的展开和折叠事件，默认值为 true
ExpandDepth	第一次显示 TreeView 控件时，树的展开层次数，默认值为 FullyExpand(即-1)，表示全部展开所有节点
ExpandImageToolTip	当用户将鼠标悬停在可展开菜单项上时显示的工具提示
Nodes	设置 TreeView 控件的各级节点及其属性
ShowExpandCollapse	是否显示折叠、展开图像，默认值为 true
ShowLines	是否显示连接子节点和父节点之间的连线，默认值为 false
ShowCheckBoxes	指示在哪些类型节点的文本前显示复选框。其共有 5 个属性值：None(所有节点均不显示)、Root(仅在根节点前显示)、Parent(仅在父节点前显示)、Leaf(仅在叶节点前显示)和 All(所有节点前均显示)。默认值为 None
HoverNodeStyle	当鼠标悬停于节点上时显示的节点样式
LeafNodeStyle	叶节点的样式
LevelStyle	特殊深度节点的样式
NodeStyle	所有节点的默认样式
ParentNodeStyle	父节点的样式
RootNodeStyle	根节点的样式
SelectedNodeStyle	选定节点的样式



下面通过一个例子演示如何利用 TreeView 控件和 SiteMapDataSource 控件实现网站导航。

【例 3-8】在例【3-5】的基础上，利用 TreeView 控件实现网站导航功能。

(1) 启动 VWD 2010，打开网站【例 3-5】。

(2) 在【解决方案资源管理器】中，添加名为 TreeView.aspx 的网页。

(3) 在该页面中添加一个 TreeView 控件，单击 TreeView 控件右上方的小三角，打开【TreeView 任务】面板，在【选择数据源】下拉列表中选择【<新建数据源>】选项，如图 3-32 所示。

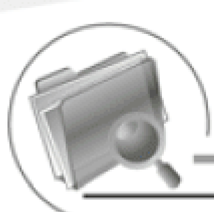




图 3-32 【TreeView 任务】面板

(4) 在打开的【数据源配置向导】对话框中选择【站点地图】选项，如图 3-33 所示。

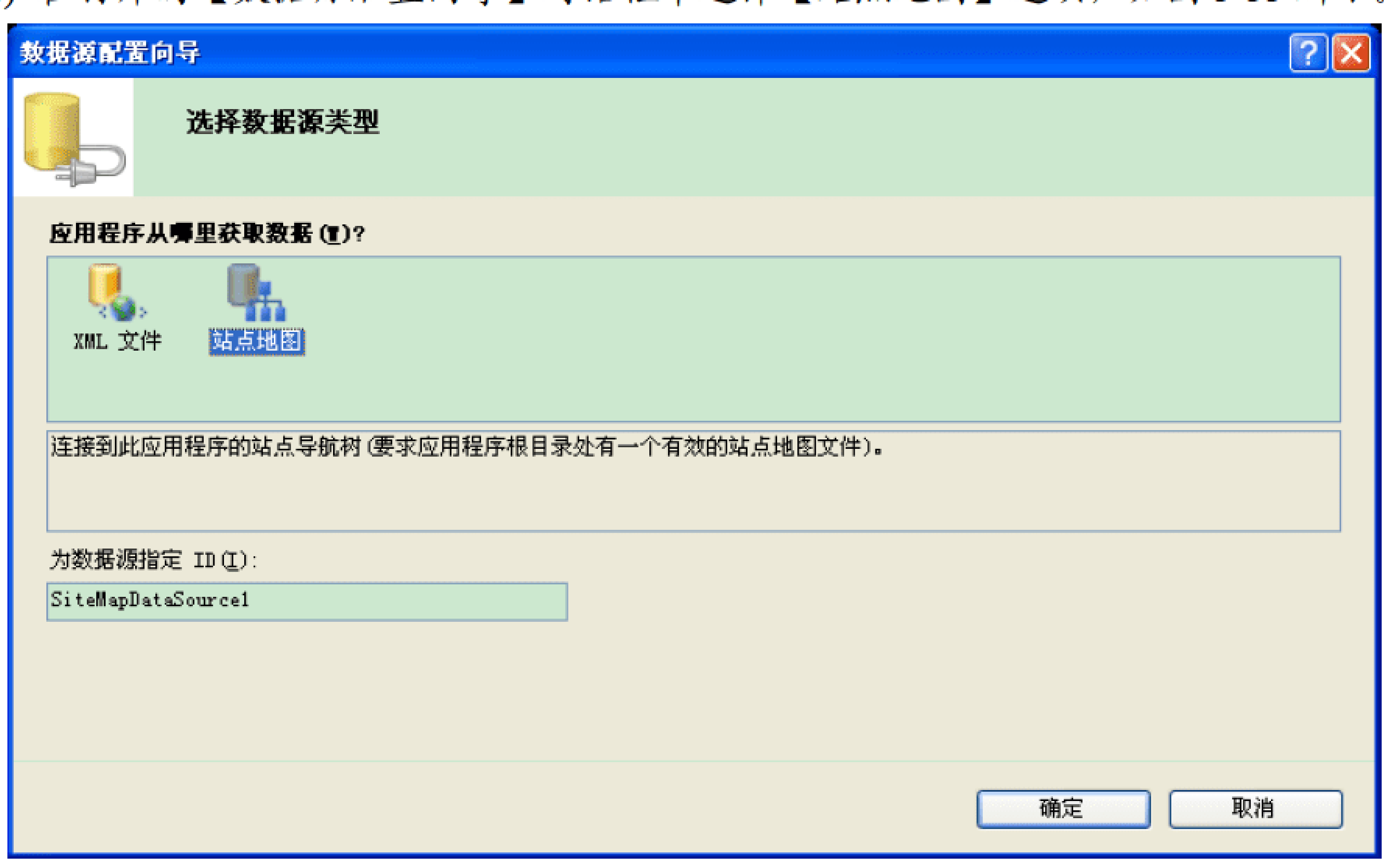


图 3-33 【数据源配置向导】对话框

(5) 单击【确定】按钮后，SiteMapDataSource 控件将从 Web.sitemap 文件中得到站点信息，此时的 TreeView 控件如图 3-34 所示。

(6) 编译并运行程序，在浏览器中加载 TreeView.aspx 页面，效果如图 3-35 所示。



图 3-34 设置数据源后的 TreeView 控件



图 3-35 TreeView 导航示例





提示

使用 TreeView 时遇到的问题与在早期 ASP.NET 版本中使用 Menu 时遇到的相同，即产生大量的 HTML 标记。但是，该控件没有 RenderingMode 属性，因此，如果使用 TreeView 控件，将会在处理基于表的 HTML 时遇到麻烦。

5. 关于导航的实用提示

下面列出了关于网站导航的一些实用提示。

- 当开始构建一个知道将来会增长的 Web 站点时，创建一个逻辑结构。不要把所有文件都放在 Web 站点的根文件夹中，而要将相关的文件根据逻辑组合到同一个文件夹中。这样的逻辑组合可以使站点容易管理，用户容易找到所需的页面。虽然用 Web.sitemap 文件很容易将页面移到 Menu 或 TreeView 中，但是如果同时使用了编程方式的重定向或转换时就较难了，因为还需要更新服务器端代码以反映新的站点结构。为了创建稳固的页面结构，可以在开始构建站点之前把它画在纸上，或者使用站点地图图表工具，如 Microsoft Visio。
- 尽量限制显示在 Menu 或 TreeView 控件中的主项和子项的数目。如果供用户选择的选项列表过长，用户会迷路或者混淆。
- 当创建用来存储页面的文件夹时，最好给其起简短且有逻辑的名称。

3.2.6　登录控件

与数据和导航控件一样，登录控件也是在 ASP.NET 2.0 中引入的，并且在 ASP.NET 4.0 中仍有着强大的功能。有了登录控件，只需很少的工作量就可以构建安全的 Web 站点，另外，ASP.NET 4.0 还提供了一些工具，允许用户修改密码；或者当用户忘记密码时，可以请求一个新的密码，并允许根据用户的登录状态和角色显示不同的数据。

ASP.NET 4.0 中包含 7 个登录控件，每个都有不同的用途。下面详细介绍每个控件的用法。

1. Login

Login 控件允许用户登录到站点。而在后台，它通过应用程序服务与配置好的成员提供者进行通信，查看用户名和密码是否代表系统中的有效用户。如果用户通过验证，就生成发送到用户浏览器的 Cookie。对于后续的请求，浏览器重新提交该 Cookie 给服务器，这样系统就知道它仍在处理有效用户。成员提供者的不同设置都在 Web.config 文件的<membership />元素中进行配置。

Login 控件的常用属性如表 3-11 所示。

表 3-11　Login 控件的常用属性

属　　性	说　　明
DestinationPageUrl	该属性定义了登录请求成功后将用户发往哪个 URL
CreateUserText	该属性控制用于邀请用户注册新帐户的文本
CreateUserUrl	该属性控制用户注册新帐户的页面的 URL
DisplayRememberMe	该属性指定控件是否显示 Remember Me 选项。如果设置为 false 或在登录时未选中该复选框，那么每次关闭和重新打开浏览器时，用户需要重新进行身份验证
RememberMeSet	该属性指定最初是否选择 Remember Me 选项
PasswordRecoveryText	该属性控制用于告诉用户可重置或恢复密码的文本
PasswordRecoveryUrl	该属性确定用户可获取新密码的页面的 URL
VisibleWhenLoggedIn	该属性控制当前用户登录时控件是否可见，默认值为 true



除了这些属性，Login 控件还有一些 Text 属性，如 LoginButtonText、RememberMeText、TitleText 和 UserNameLabelText，这些属性用于设置控件中和其子控件(如组成用户界面的 Button 和 Label 控件)上出现的文本信息。

默认情况下，ASP.NET 的身份验证机制假定在站点的根目录下有一个用于用户登录的页面 Login.aspx。这个页面要想生效，最少需要有一个 Login 控件。如果要使用不同的页面，可以在 <authentication />下面的<forms />元素中指定其路径，如下所示：

```
<authentication mode="Forms">
  <forms loginUrl="MyLoginPage.aspx" />
</authentication>
```



提示

在 Login 页面(配置在 loginUrl 中)上，Login 控件的 VisibleWhenLoggedIn 属性没有任何作用。在配置过的 Login 页面上，Login 控件总是可见。如果想要隐藏它，可以使用 LoginView 控件。

Login 控件也提供了一些事件，这些事件通常不需要进行处理，但经常会派上用场。例如，LoggedIn 事件在用户刚登录后触发，如果 DestinationPageUrl 不太灵活，这里是将用户动态发送到另一页面的理想场所。

2. LoginView

LoginView 控件可用于向不同的用户显示不同的数据。该控件可以区分匿名用户和登录用户，甚至区分不同角色中的用户。LoginView 是模板驱动的，因此可允许定义显示给不同用户的不同模板。如表 3-12 所示列出了两个主要的模板和特殊的 RoleGroups 元素。



表 3-12　LoginView 控件的主要模板和 RoleGroups 元素

模　板	说　明
AnonymousTemplate	该模板中的内容只显示给未进行身份验证的用户
LoggedInTemplate	该模板中的内容只显示给登录用户。该模板与 AnonymousTemplate 互斥。任何时候只有其中一个模板可见
RoleGroups	该模板可以包含一个或多个 RoleGroup 元素，这些元素包含一个定义特定角色的内容的 ContentTemplate 元素。允许查看内容的角色定义在 Roles 属性中，该属性采用一个逗号分隔的角色列表。RoleGroups 元素与 LoggedInTemplate 互斥，这意味着如果用户是 RoleGroup 的其中一个角色中的成员，那么 LoggedInTemplate 中的内容就不可见。另外，只有匹配用户角色的第一个 RoleGroup 的内容可见

除了定义在控件各种子元素中的内容之外，LoginView 控件本身并不输出任何标记，这意味着可以很容易地将它嵌入一对 HTML 标记之间，如<h1>……</h1>和
……，从而创建自定义标题或者列表项。



3. LoginStatus

LoginStatus 控件提供了有关用户当前状态的信息。当用户未进行身份验证时，它提供【登录】链接；当用户登录后，它提供【注销】链接。通过设置 LoginText 和 LogoutText 属性，可以控制实际显示的文本；也可以设置 LoginImageUrl 和 LogoutImageUrl 属性显示图像而非文本；LogoutAction 属性可用来决定在用户注销时是否刷新当前页面，或是否在用户注销后将用户带至另一页面，通过设置 LogoutPageUrl 可以确定这一目标页面。

除了这些属性，该控件可以引发两个事件：LoggingOut 和 LoggedOut，它们分别在用户刚注销前后触发。

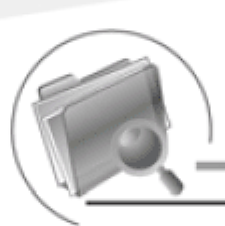
4. LoginName

LoginName 是一个极为简单的控件。它所做的就是显示登录用户的名称。为了将用户名嵌入到一些文本中，可以使用 FormatString 属性。如将 FormatString 属性设置为“当前登录用户是：{0}”，则运行时{0}将被用户名所取代。

5. CreateUserWizard

除了允许用户登录和使用当前用户的登录状态显示或隐藏相关内容的控件外，登录类控件中还包含另外 3 个控件，它们允许用户在站点上注册新用户、修改现有密码或是恢复丢失密码。下面将介绍这些控件。

CreateUserWizard 控件用于注册新用户，该控件有一个较长的 Text 属性列表，包括 CancelButtonText、CompleteSuccessText、UserName- LabelText 和 CreateUserButtonText，它们会影响控件中使用的文本。所有属性都有默认设置，开发人员也可以将其修改为满足自己需求的内容。



除了文本类属性，该控件还有许多以 ImageUrl 结尾的属性，如 CreateUserButtonImageUrl。这些属性允许定义各种用户动作的图像而非控件生成的默认按钮。如果设置任一属性为有效的 ImageUrl，则还需要设置相应的 ButtonType。例如，要将【创建用户】按钮修改为图像，则需要设置 CreateUserButtonImageUrl 为有效图像并设置 CreateUserButtonType 为 Image。ButtonType 的默认值为 Button，默认情况下它呈现为标准的按钮。也可以设置这些属性为 Link，这样它们将呈现为标准的 LinkButton 控件。

另外，该控件还提供了以下可设置用于修改其行为和外观的有用属性。

- ContinueDestinationPageUrl：该属性定义用户在注册后单击【继续】按钮时被带往的页面。
- DisableCreatedUser：当帐户创建时是否将用户标记为禁用的。如果设置为 True，那么在帐户被启用前，用户不能登录到该站点。默认为 False。
- LoginCreatedUser：在帐户创建后是否让用户自动登录，默认为 True。
- RequireEmail：决定控件是否要求用户提供电子邮件地址，默认为 True。
- MailDefinition：该属性包含大量子属性，允许定义在用户注册后发送给用户的电子邮件。



CreateUserWizard 控件能够向用户发送一封确认电子邮件，通知用户新帐户已经成功建立。这封电子邮件可以告知其用户名和密码。

6. PasswordRecovery

PasswordRecovery 控件允许用户获得自己已有的密码(如果系统支持)或是获得一个新的自动生成的密码。在这两种情况下，密码都将发送到用户注册时输入的电子邮件地址中。

PasswordRecovery 控件的大部分属性都是我们所熟悉的。它有大量 Text 属性，如 GeneralFailureText (当密码不可恢复时显示)和 SuccessText，该属性允许设置控件显示的文本。还有一些以 ButtonType、ButtonText 和 ButtonImageUrl 结尾的属性，这些属性允许修改控件的不同动作按钮的外观和行为。如果密码成功恢复，可以设置 SuccessPageUrl 将用户导航到另一个页面。

和 CreateUserWizard 一样，PasswordRecovery 也有一个 MailDefinition 元素，该元素用于指向作为邮件正文发送的文件。可以对用户名和密码使用同样的占位符来自定义消息。如果不对 MailDefinition 进行配置，则控件会使用一个默认的邮件正文。

7. ChangePassword

ChangePassword 控件允许已有用户和登录用户更改其密码。类似于 CreateUserWizard 和 PasswordRecovery 控件，它有许多属性，可用于修改文本、错误消息和按钮。它也有一个 MailDefinition 元素，允许发送新密码的确认消息给用户的电子邮件地址。

8. 使用登录控件

本例将演示登录控件的使用，在本例创建的网站中包括登录、注册新用户、更改密码、忘记密码等与网站登录相关的全部功能。

【例 3-9】新建网站，利用登录控件实现网站登录相关的基本功能。



(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-9】。

(2) 在应用程序中分别添加名为 Login.aspx、Register.aspx、ChangePassword.aspx、PasswordRecovery.aspx 的网页。

(3) 在 Default.aspx 的设计视图中，添加一个 LoginView 控件和一个 LoginStatus 控件。

(4) 切换到源视图，LoginStatus 控件的属性保持默认设置不变，在 LoginView 控件中，添加如下代码:

```
<asp:LoginView ID="LoginView1" runat="server">
<AnonymousTemplate>
这是匿名用户看到的信息，是否<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Register.aspx">
注册</asp:HyperLink>为小石头网站的新用户。
</AnonymousTemplate>
<LoggedInTemplate>
这是登录会员看到的信息，
<asp:LoginName ID="LoginName1" runat="server" FormatString="欢迎你，{0}" />
<br />
你可以
    <asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="~/ChangePassword.aspx">修改密码
</asp:HyperLink>
</LoggedInTemplate>
</asp:LoginView>
```

上述代码通过模板区分了匿名用户和登录用户看到的信息，在匿名用户区域添加了【注册】链接；在登录用户区域使用了 Login 控件显示登录用户信息，并使用 HyperLink 控件提供了【修改密码】链接。

(5) 如果尚未登录，LoginStatus 控件将提供【登录】链接，但是如何让这个【登录】链接跳转到 Login.axpx 页面呢？这需要修改 Web.config 的配置信息。打开 Web.config，添加或修改认证模式为如下代码:

```
<authentication mode="Forms">
  <forms loginUrl="Login.aspx" timeout="2880" />
</authentication>
```

(6) 在 Login.aspx 页面中添加一个 Login 控件，设置控件的 CreateUserText 属性为“注册新用户”，CreateUserUrl 属性为“~/Register.aspx”，PasswordRecoveryText 属性为“忘记密码”，PasswordRecoveryUrl 属性为“~/PasswordRecovery.aspx”。

(7) 为了在注册新用户成功以后和用户忘记密码时能够给用户发送电子邮件，还需要进行以下设置。在 Register.aspx 页面中添加一个 CreateUserWizard 控件，设置控件的 ContinueDestinationPageUrl 属性为“~/Default.aspx”，使得创建用户成功后，单击【继续】按钮可以导航到 Default.aspx 页面。

(8) 为了在注册新用户成功以后和用户忘记密码时能够给用户发送电子邮件，所以还需要进行以下设置。



(9) 添加一个名为 RegistConfirm.txt 的文本文件到 App_Data 文件夹中，文件内容如下:

```
Hi <% UserName %>,

感谢你注册为小石头网站的新用户.
请牢记以下信息:
用户名:    <% UserName %>
密    码:    <% Password %>

小石头网站全体人员祝你身体健康
```



知识点

在输入 UserName 和 Password 占位符时要注意，它们被包含在一对服务器端标记(<%和%>)中，从而被赋予特殊的意思。



(10) 回到 Register.aspx 页面，选中 CreateUserWizard 控件，在【属性】面板中定位到 MailDefinition 属性并展开它。设置 BodyFileName 属性为刚才创建的 RegistConfirm.txt，设置 Subject 属性为“你是小石头网站的新用户”。

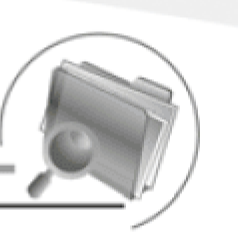
(11) MailDefinition 属性设置好了，要保证邮件能够正确发送，还需要在 Web.config 文件中配置邮件服务器，再次打开 Web.config 文件，在根元素<configuration>中添加或修改<system.net>元素，此处需要知道 SMTP 服务器的地址和发送邮件的 Email 地址:

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network" from="Your Name &lt;you@163.com&gt;">
      <network host="smtp.163.com" />
    </smtp>
  </mailSettings>
</system.net>
```

如果 ISP 要求在发送电子邮件之前进行身份验证或者希望使用一个不同的端口号，可以向<network />元素中添加如下信息:

```
<smtp deliveryMethod="Network">
  <network host="smtp.domain.com" userName="UserName" password="Password"
          port="587" />
</smtp>
```

一些邮件服务器要求使用 SSL。SSL 是一种加密技术，用于加密发送到邮件服务器的数据以提高安全性能。在 ASP.NET 4.0 之前的版本中，必须在代码中编程激活 SSL。而在 ASP.NET 4.0



中，<network />元素上有了 enableSsl 特性，要使用需要支持 SSL 的邮件服务器，可以使用如下所示的<network />元素：

```
<network enableSsl="true" host="smtp.gmail.com" password="Password" userName="you@gmail.com" />
```

(12) 在 ChangePassword.aspx 页面中添加一个 ChangePassword 控件，设置控件的 ContinueDestinationPageUrl 和 CancelDestinationPageUrl 属性均为“~/Default.aspx”，即密码修改成功或取消密码修改都将导航到 Default.aspx 页面。

(13) 在 PasswordRecovery.aspx 页面添加一个 PasswordRecovery 控件，设置控件的 MailDefinitionSubject 属性为“你的小石头网站的新密码”。当然，此处也可以设置一个邮件模板，类似于 CreateUserWizard 控件的设置一样。

(14) 至此，网站建设工作已基本完成，这时就可以编译并运行程序，在浏览器中访问网站了。首先，匿名用户或未登录用户看到的 Default.aspx 页面如图 3-36 所示。

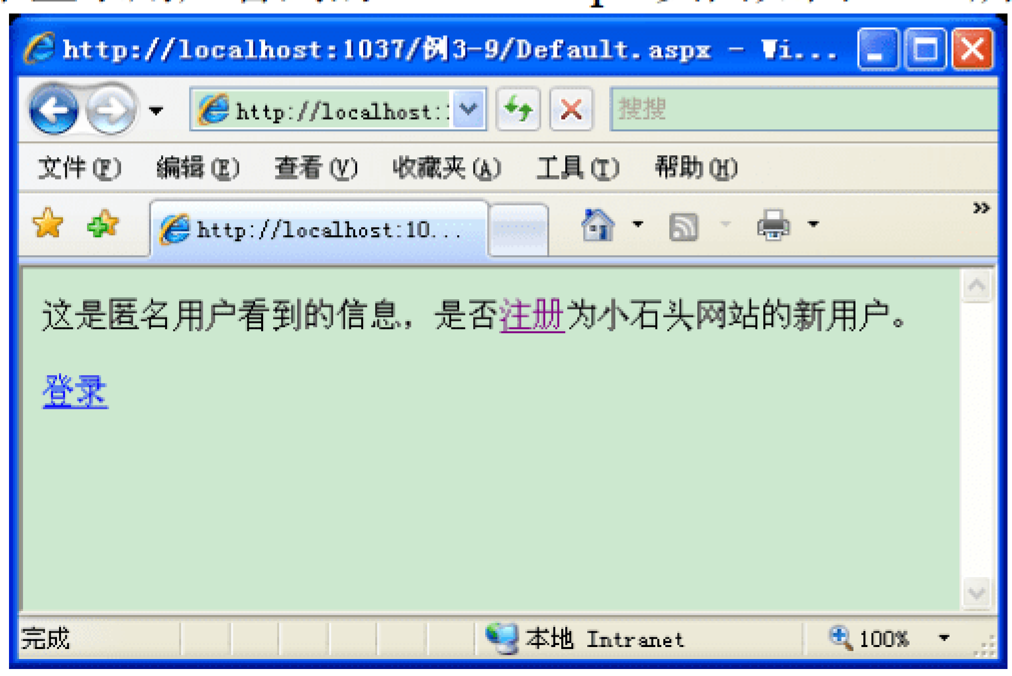


图 3-36　匿名或未登录用户看到的页面

单击【注册】链接进入注册用户页面，在此输入注册信息，如图 3-37 所示，单击【创建用户】按钮，提示创建成功，如图 3-38 所示。

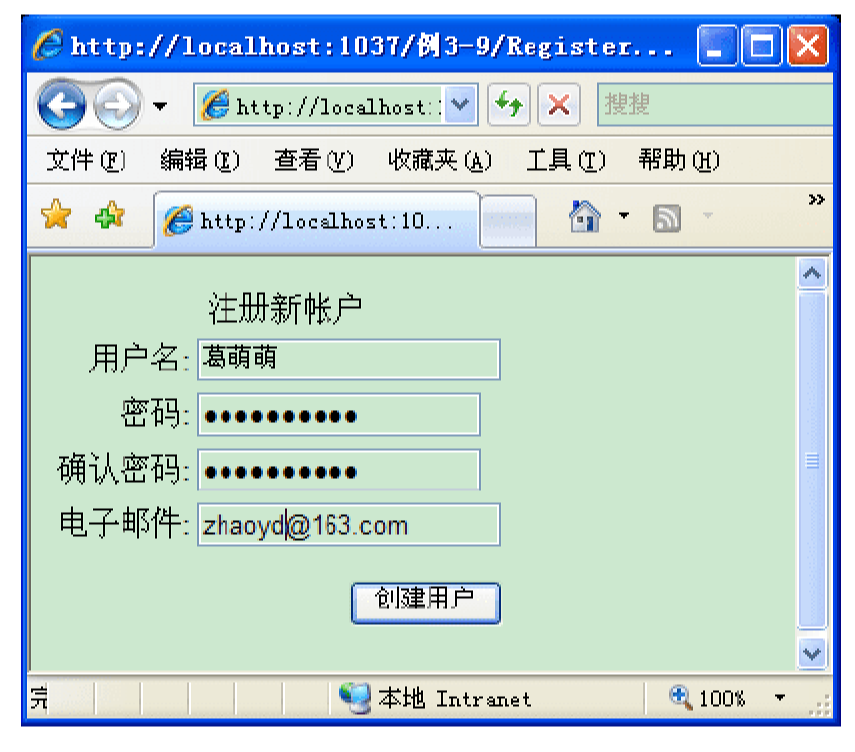


图 3-37　注册页面

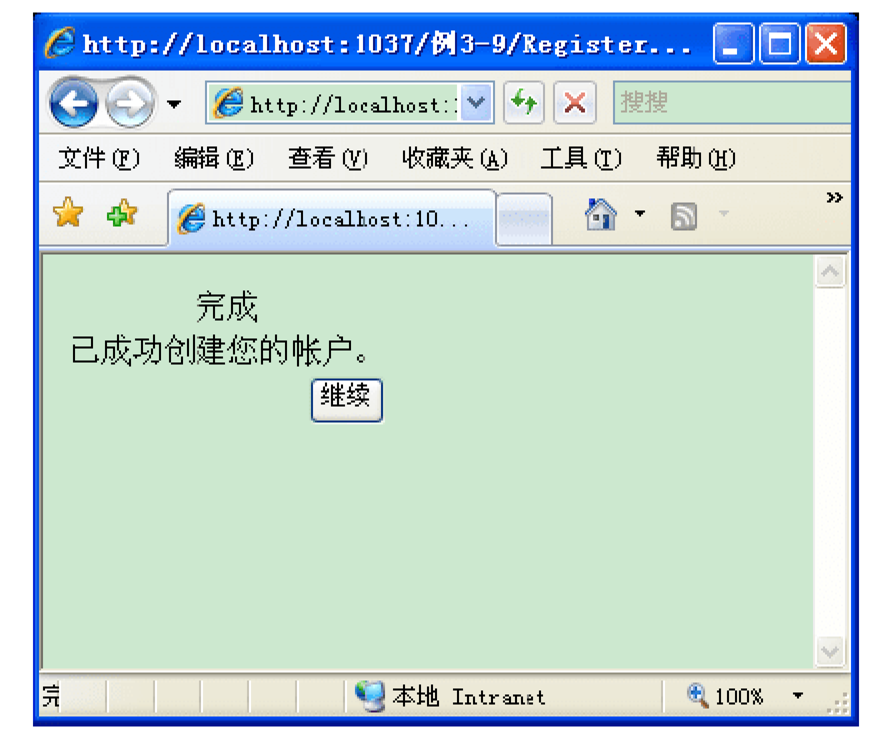


图 3-38　注册成功页面

读者也许再问，注册的用户信息保存在什么地方了呢？在创建网站的过程中并没有处理这些信息啊？其实，这里还有一个应用程序的配置没有介绍。在首次尝试登录(或使用需要数据库



访问的其他登录控件)时，提供者检查应用程序是否在使用带有必要数据库对象(如表)的数据库。默认情况下，它通过查找名为 LocalSqlServer 的连接字符串检查数据库。在第 2 章介绍的 Web.config 文件中，无法找到这一连接字符串，因为它定义在.NET Framework 文件夹(C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\Config)中名为 machine.config 的文件中。该文件中的连接字符串类似如下：

```
<connectionStrings>
    <add name="LocalSqlServer" connectionString="data source=.\SQLEXPRESS;
            Integrated Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;
            User Instance=true"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```

这是一个针对 SQL Server 2008 Express Edition(Microsoft SQL Server 的免费版本)的连接字符串。可以使用|DataDirectory|的连接字符串指向 Web 站点的 App_Data 文件夹中的数据库。如果首次使用某个登录控件或其他应用程序服务时，ASPNETDB.MDF 数据库并没有在特定位置出现，那么应用程序服务就会自动创建一个 ASP.NETDB.MDF 数据库，所以在第一次注册或者没注册直接登录时会感觉到慢。

此时，可返回到 VWD 中，单击【解决方案资源管理器】面板中的 App_Data 文件夹，并单击该窗口工具栏中的刷新按钮，将看到如图 3-39 所示的新数据库 ASPNETDB.MDF。

数据这个数据库文件将在【数据库资源管理器】面板中打开它。展开【表】节点，可以看到已添加的表，如图 3-40 所示。



图 3-39 【解决方案资源管理器】面板

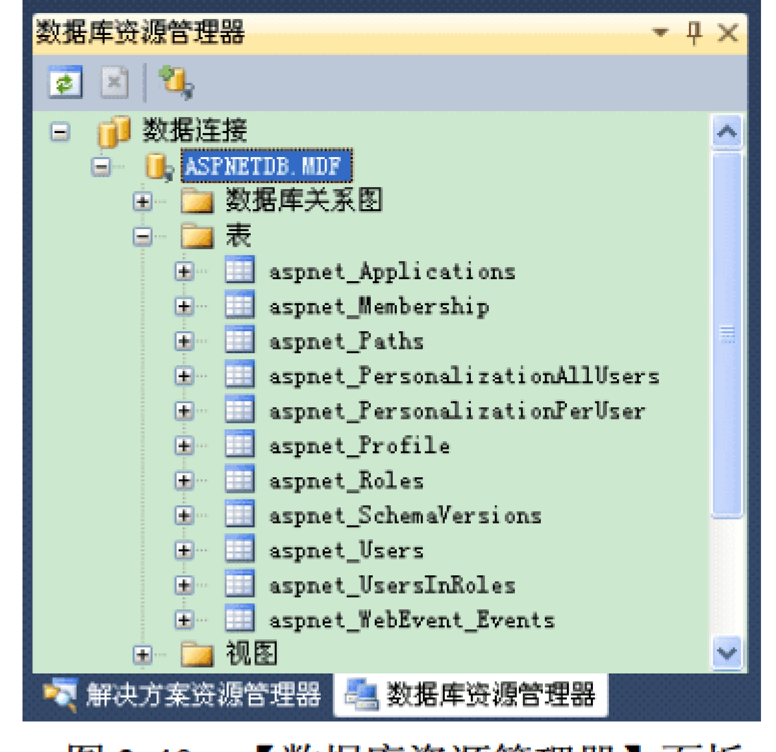
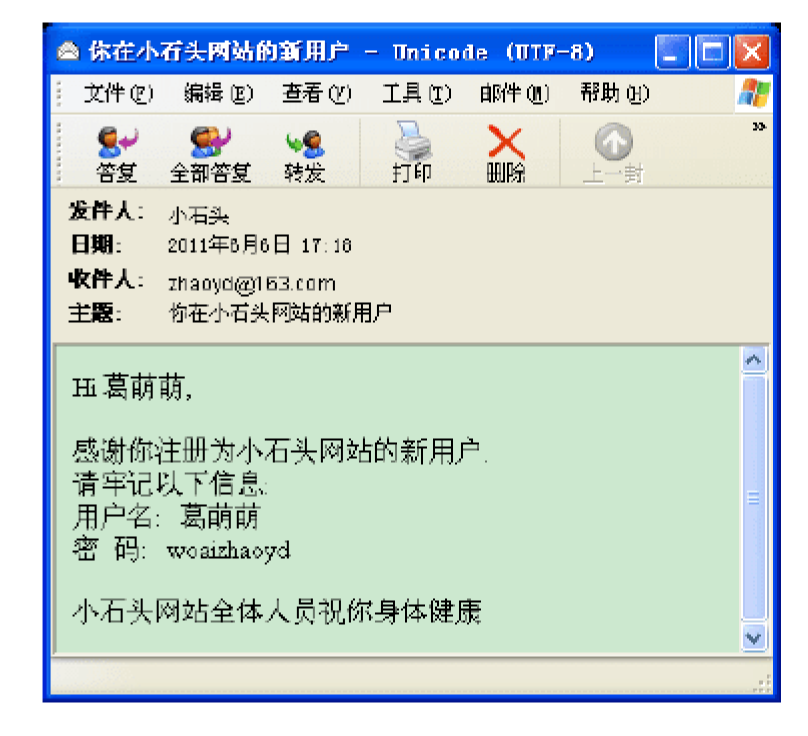


图 3-40 【数据库资源管理器】面板

在成功创建数据库后，登录控件就可以使用它。例如，在使用 CreateUserWizard 控件创建一个新帐户时，记录就插入到 aspnet_Membership 和 aspnet_Users 表中。

注册成功后，注册时填写的 Email 中将收到一封由网站发出的邮件，邮件内容就是前面设置的 RegistConfirm.txt 模板，如图 3-41 所示是收到邮件。

单击【继续】按钮将返回到 Default.aspx 页面，此时看到的页面是登录用户所看到的信息，如图 3-42 所示。



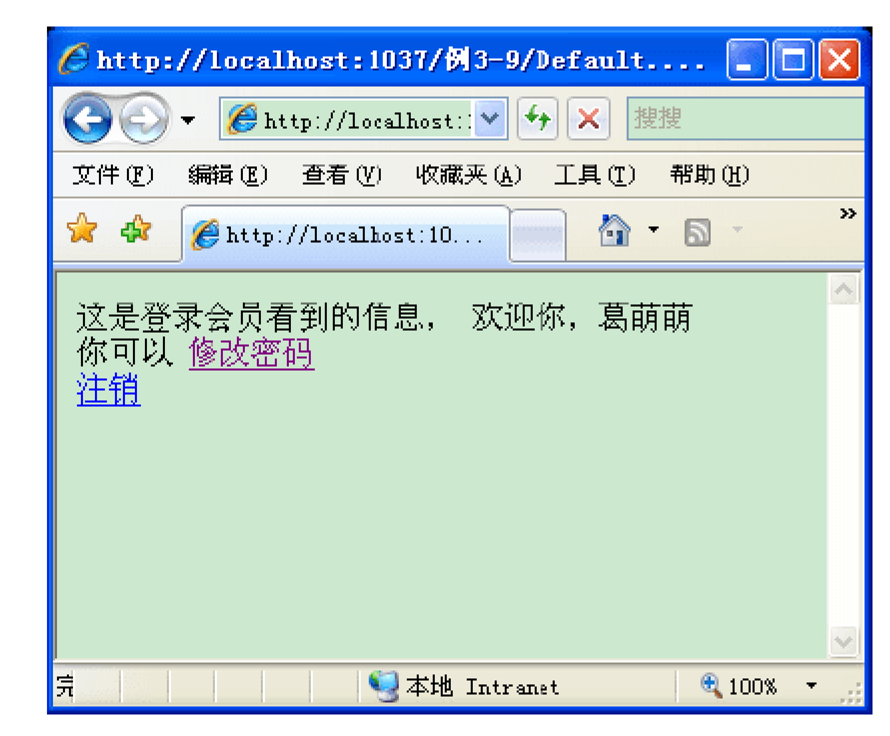
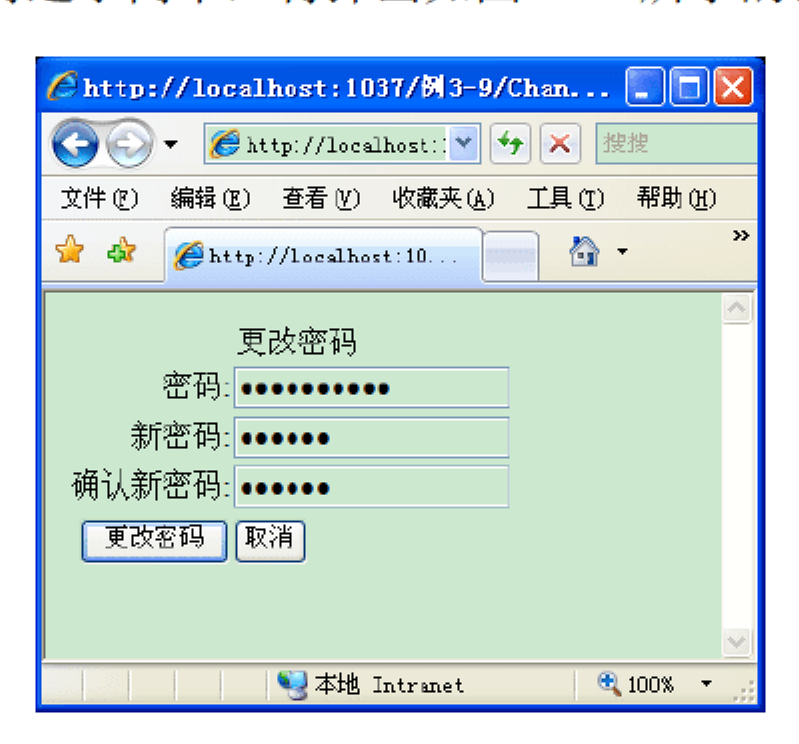



图 3-41　网站发出的注册成功邮件　　　图 3-42　登录用户看到的信息

此时可以【修改密码】，也可以【注销】，然后重新登录。单击【修改密码】链接，进入 ChangePassword.aspx 页面，如图 3-43 所示。在此输入新旧密码，单击【更改密码】按钮，如果密码设置得过于简单，将弹出如图 3-44 所示的错误提示。



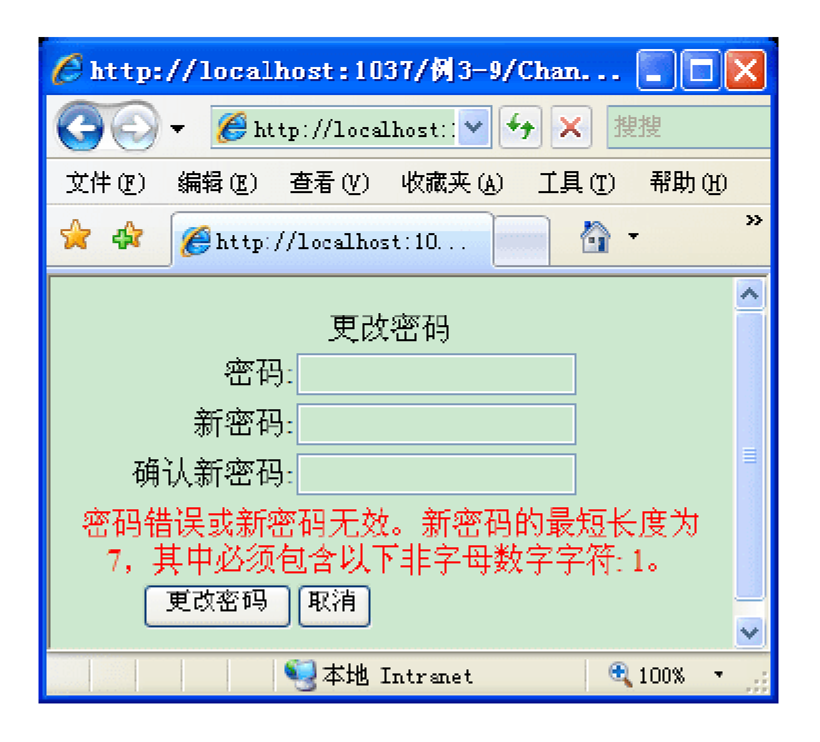


图 3-43　更该密码页面　　　图 3-44　更改密码错误提示

这是因为登录控件使用了一些默认属性。其实在注册新用户时，如果密码过于简单也同样会报类似的错误。下面就来介绍这些默认配置的来源，以及如何来修改这个配置。

在 C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\Config 目录找到 machine.config 文件，并用记事本将其打开，定位到<system.web>下面的<membership>元素，如下所示：

```
<membership>
    <providers>
        <add name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider,
System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a" connectionStringName=
"LocalSqlServer" enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="true"
applicationName="/" requiresUniqueEmail="false" passwordFormat="Hashed" maxInvalidPasswordAttempts="5"
minRequiredPasswordLength="7" minRequiredNonalphanumericCharacters="1" passwordAttemptWindow="10"
passwordStrengthRegularExpression=""/>
```




```
        </providers>
    </membership>
```

这就是 ASP.NET 默认的安全策略配置。如果要为某个应用程序配置不同的安全策略，可以在应用程序的 Web.config 文件中修改这些设置。

返回到 VWD 2010，打开 Web.config 文件，在<system.web>下面添加如下<membership>元素：

```
    <membership>
      <providers>
        <clear/>
        <add name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider"
connectionStringName="ApplicationServices"
                enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="false"
requiresUniqueEmail="false"
                maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6"
minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
                applicationName="/" />
      </providers>
    </membership>
```

这样就修改为密码至少 6 个字符，并且可以没有特殊字符了。

默认情况下，密码在数据库中是以散列形式存储，散列是一个不可逆的进程，它为数据创建唯一索引。因为它是不可逆的，所以没有办法通过散列重新得到原始密码，这使得密码可以更安全地存储在数据库中。而对于忘记密码的请客，PasswordRecovery 控件将生成新密码，然后将新密码发送到与用户注册时填写的 Email 地址中。如图 3-45 所示为发送新密码的邮件。



图 3-45　发送新密码的邮件

3.2.7　WebParts

ASP.NET　WebParts 是一组控件，允许 Web 页面的终端用户修改 Web 站点的外观和行为。



用户只需通过几个简单的动作，就能修改 Web 站点的整个外观。这些动作包括：重新排列内容、隐藏或显示部分 Web 页面，以及向页面中添加其他内容片段。ASP.NET WebParts 超出了本书的讨论范围，因为只是介绍它就需要一整本书。如果要了解关于 WebParts 的更多内容，可以参考其他相关资料和网站。

3.2.8　Ajax Extensions

在 2005 年 11 月官方发布 ASP.NET 2.0 一年多后，Microsoft 发布了 ASP.NET 2.0 AJAX Extensions 1.0 作为 ASP.NET 2.0 的插件。有了这些扩展，可以创建无闪烁的 Web 应用程序，且不需要完整回发就能从客户端 JavaScript 中检索服务器上的数据。自从 Ajax 在 2005 年成为一种热门技术以来，Microsoft 一直致力于成为顶级 Ajax 的实现者。Ajax 扩展现在已经完全集成到了 VWD 2008 IDE 中，并在 VWD 2010 中升级为 AJAX 4。本书第 7 章将重点介绍 Ajax。



3.2.9　动态数据

这种类别的控件用于动态数据 Web 站点。动态数据站点允许在数据库中快速创建用户界面来管理数据。本书不对这些控件作详细讨论，感兴趣的读者可参考其他数据或 VWD 在线帮助。

3.3　用户控件

有时可能需要控件具有 ASP.NET 内置服务器控件没有的功能。在这种情况下，用户可以创建自己的控件。有两个选择，可以创建用户控件和自定义控件。

用户控件是能够在其中放置标记和服务器控件的容器。然后，可以将用户控件作为一个单元对待，为其定义属性和方法。

自定义控件是编写的一个类，此类从 Control 或 WebControl 派生。

创建用户控件要比创建自定义控件方便很多，因为可以重用现有的控件。用户控件使创建具有复杂用户界面元素的控件极为方便。

3.3.1　用户控件简介

用户控件对于封装需要在整个站点中重复使用的标记、控件和代码来说非常有用。在某种程度上，用户控件看起来有一些像服务器控件，它们都可以包含能够在页面中重用的编程逻辑和表现。

从开发设计角度来看，用户控件与完整的 ASP.NET 网页(.aspx 文件)也非常相似，同时具有用户界面页和代码页。可以采取与创建 ASP.NET 页相似的方式创建用户控件，然后向其中添加所需的标记和子控件。用户控件可以像页面一样包含对其内容进行操作的代码。

但是，用户控件与 ASP.NET 网页也有以下一些区别：

- 用户控件的文件扩展名为.ascx。
- 用户控件中没有@Page 指令，而是包含@Control 指令，该指令对配置及其他属性进行定义。
- 用户控件不能作为独立文件运行。而必须像处理任何控件一样，将它们添加到 ASP.NET 页面中。
- 用户控件中没有 HTML、body 或 form 元素。这些元素必须位于宿主页中。
- 可以在用户控件上使用与在 ASP.NET 网页上所用相同的 HTML 元素(HTML、body 或 form 元素除外)和 Web 控件。例如，如果要创建一个将用作工具栏的用户控件，则可以将一系列 Button 服务器控件放在该控件上，并创建这些按钮的事件处理程序。

3.3.2 创建并使用用户控件

本节将来介绍如何创建并使用用户控件，这里创建的是一个实现微调控件的用户控件。

1. 创建用户控件

【例 3-10】创建一个实现微调控件的用户控件。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-10】。

(2) 在【解决方案资源管理器】面板中，右击【例 3-10】解决方案，从弹出的快捷菜单中选择【添加新项】命令，在打开的【添加新项】对话框中选择【Web 用户控件】选项，让默认名称为 WebUserControl.ascx，单击【添加】按钮。

(3) 切换到 WebUserControl.ascx 设计视图，添加 1 个 TextBox 控件和 2 个 Button 控件到设计窗口中。两个 Button 控件 Text 属性分别设置为 “<” 和 “>”，TextBox 控件的 Text 属性为 “0”，控件布局如图 3-46 所示。

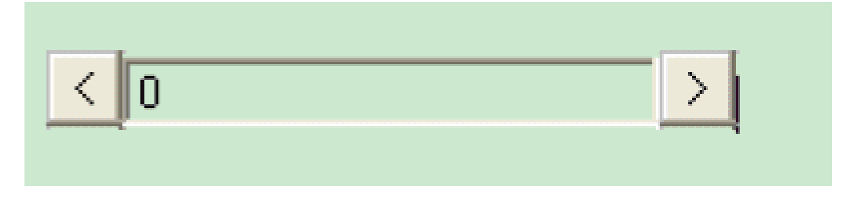


图 3-46　控件布局

(4) 切换到 WebUserControl1 的源视图，即打开 WebUserControl.ascx.cs 文件，定义变量，为 Page_Load 事件、两个按钮的单击事件添加如下代码:

```
protected int count;
protected void Page_Load(object sender, EventArgs e)
{
```



```
        if (IsPostBack)
        {
            count = int.Parse( TextBox1.Text);
        }
        else
        {
            count = 0;
        }
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        count++;
        TextBox1.Text = count.ToString();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        count--;
        TextBox1.Text = count.ToString();
    }
```

(5) 保存用户控件，至此完成用户控件的创建，但这时还不能在浏览器中查看该控件的效果，而必须将该用户控件添加到 ASP.NET 页面中才行。

2. 使用用户控件

要在 ASP.NET 页面或另一个用户控件中使用一个用户控件，需要执行如下两个步骤。

(1) 需要注册控件，方法是向希望出现用户控件的页面或控件中添加一个@ Register 指令。

(2) 向页面添加用户控件的标记，并可以(可选地)在其上设置一些特性。

@ Register 指令包含以下 3 个重要的特性。

- src：指向要使用的用户控件。为了在以后的阶段使页面的移动更容易，也可以用~语法指向应用程序根文件夹中的控件。
- tagname：用在页面的控件声明中的标记名。可以自由地命名这个名称，但通常都让它与控件的名称相同。
- tagprefix：容纳用在页面的控件声明中的 TagName 的前缀。正如 ASP.NET 用 asp 前缀指代它的控件一样，也需要为自己的用户控件提供一个前缀。默认情况下，这个前缀是 uc，后跟一个序号，不过也可以将它改为其他内容，如，改为公司名称或者自定义的缩略词。

【例 3-11】使用【例 3-10】中创建的实现微调控件的用户控件。

(1) 启动 VWD 2010，选择【文件】|【打开网站】命令，打开网站【例 3-10】。



(2) 切换到 Default.aspx 的设计视图，从【解决方案资源管理器】面板中拖动 WebUserControl.ascx 到页面中。

(3) 切换到源视图，可以看到在@Page 指令下方自动生成的@Register 指令，如下：

```
<%@ Register src="WebUserControl.ascx" tagname="WebUserControl" tagprefix="uc1" %>
```

<div>中生成的声明控件的代码如下：

```
<div>
    <uc1:WebUserControl ID="WebUserControl1" runat="server" />
</div>
```

(4) 编译并运行程序，在默认浏览器中打开 Default.aspx 页面，单击微调按钮即可改变文本框中的数值，如图 3-47 所示。

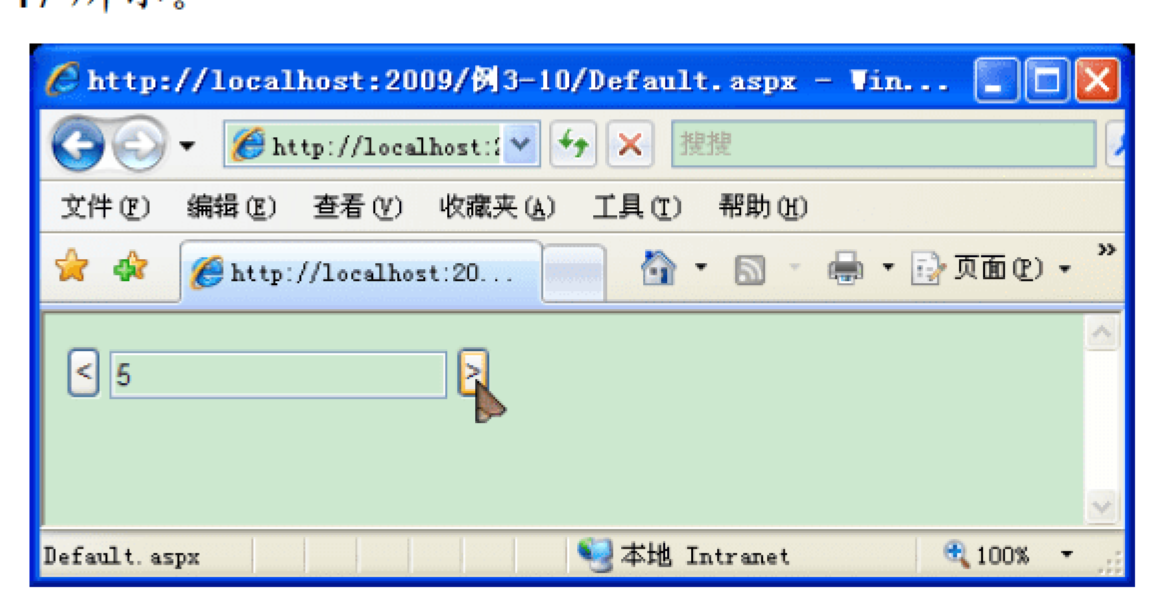


图 3-47　使用用户控件页面效果

3. 用户控件的站点范围注册

如果有一个期望在站点的单独内容页上使用得非常频繁的控件，则可以在 Web.config 文件中全局地注册这个控件。这样，它就会变得在整个站点内可用，而不需要在每个页面上注册。

在 Web.config 文件中注册用户控件的操作步骤如下：

(1) 打开站点根文件夹中的 Web.config 文件。

(2) 在<system.web>元素内添加如下代码，其中包含一个带有<add />子元素的<controls />元素：

```
<pages theme="Monochrome">
  <controls>
    <add tagPrefix="uc1" tagName="WebUserControl" src="~/WebUserControl.ascx" />
  </controls>
</pages>
```

(3) 保存修改并关闭文件。

(4) 此时，再在 ASP.NET 网页中添加用户控件时，就不需要@Register 指令了。





知识点

如果在 ASP.NET 页中没有@ Register 指令，ASP.NET 运行库就会扫描 Web.config 文件寻找在那里注册的控件，如果找到相应地用户控件的注册记录，就会将它的内容添加到页面中。

用户控件对于封装重复内容非常有效，但是它们也会使站点难以管理，因为代码和逻辑包含在多个文件中。所以，不要过度使用用户控件。如果不确定有些内容是否会在站点的另一部分中重用，可以先直接把它嵌在页面中。如果有需要，在将它移到单独的用户控件中即可。

3.4　ASP.NET 状态引擎

在第 2 章中曾介绍过视图状态对象，它是 Page 对象的一个属性，是状态管理中常用的一种对象，可以用来保存页和控件的值。本节将详细介绍状态引擎的工作原理以及如何关闭视图状态。



3.4.1　状态引擎的工作原理

在学习服务器控件时，当在浏览器中请求包含 TextBox 控件的页面，在控件中输入一些文本，并单击按钮提交时，会导致发向服务器的一个回发，当重新加载页面时，该文本仍然会出现在文本框中。如果熟悉其他 Web 技术，如 ASP 或 PHP，那么就会知道该功能的强大之处。在那些语言中，常常需要编写很多代码才能实现这个功能。那么，在 ASP.NET 中为什么会实现这种功能？它是如何自动发生的呢？

在设计时 HTTP(用来在 Web 浏览器中请求和服务页面的协议)是无状态的。就 Web 服务器而言，从浏览器中对服务器发出的页面请求和单击链接到其他页面的请求仍然是原来的意思。Web 服务器不会记忆以前请求的页面。

文本框中的文本是由 ASP.NET 状态引擎维护的，这是一个完全集成在 ASP.NET 运行库中的功能。它启用控件来维护它们跨回发的状态，因此在页面的每个回发之后它们的值和设置仍然是可用的。

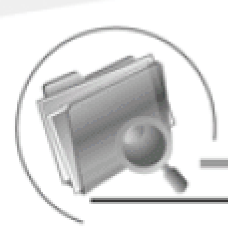
ASP.NET 中的状态引擎可以存储很多控件的状态。它不仅能存储用户输入控件(如 TextBox 和 CheckBox)的状态，而且可以存储其他控件(如 Label，甚至是 Calendar)的状态。下面来看一个具体的示例。

【例 3-12】ASP.NET 维护状态的方式。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 3-12】。

(2) 切换到设计视图，选择【表】|【插入表】命令，插入一个 2 行 2 列的表。

(3) 在第一行的第一个单元格中，添加一个 Label 控件。在第二行的第一个单元格中，添加一个 Calendar 控件。



(4) 一旦把日历控件放入单元格中，Calendar 控件的【任务】面板就会弹出，如图 3-48 所示。这个面板上只有一个选项：【自动套用格式】，它允许修改日历的外观。单击该链接，从预先定义的配色方案中选择一种，如【彩色型 2】，并单击【确定】按钮。

(5) 在右侧的两个单元格中各添加一个 Button 控件。Text 属性分别设置为“设置时间”和“提交”，如图 3-49 所示。

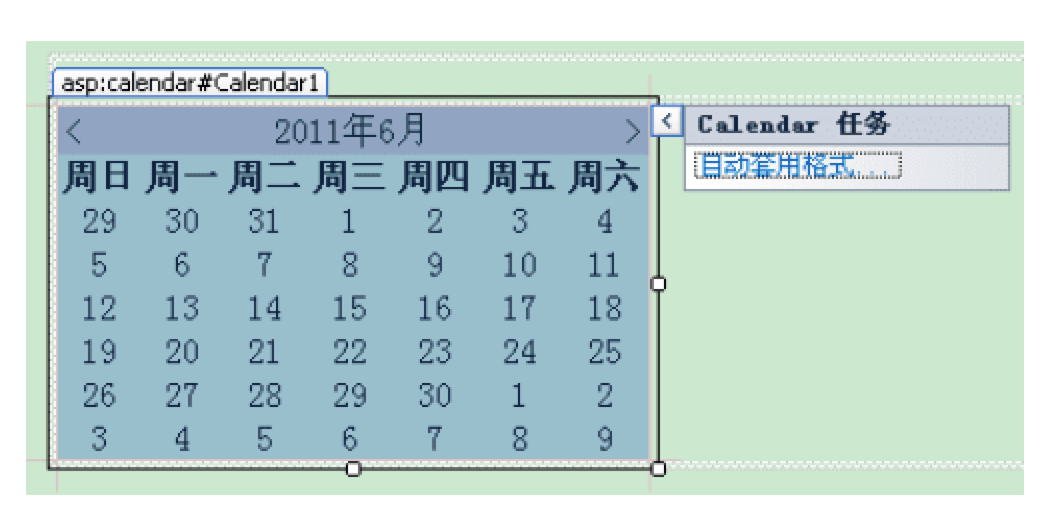





图 3-48　添加 Calendar 控件　　图 3-49　控件布局



(6) 为【设置时间】按钮添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString();
}
```

(7) 编译并运行程序，在浏览器中打开该页面。单击某一天以在日历上选择一个日期。注意，单击了日期后，页面可能会重新加载，这是由一个回发引起的。

(8) 单击几次【设置时间】按钮。页面会再次发送回服务器，并且每次单击该按钮时 Label 控件都会更新为最新的日期和时间。等待几秒钟，然后单击【提交】按钮，这时又发生了一个回发，然而 Label 控件中仍然显示的是原来的日期和时间，并没有任何变化。

(9) 回到 VWD，在设计视图中选中 Label 控件，在【属性】面板中设置 EnableViewState 属性为 False。

(10) 在浏览器中再次打开页面，重复前面的步骤，单击日历和按钮。这次，当单击【提交】按钮时或在日历上选择一个日期后，将看到 Label 控件显示为默认的初始文本 “Label” 。

为了更好地理解它的工作原理，再次在浏览器中打开页面，然后查看它的源代码。可以看到如下的<form>元素。

```
<form name="form1" method="post" action="Default.aspx" id="form1">
...
</form>
```

HTML <form>元素用于让用户从浏览器向服务器提交信息。表单的提交方式有两种：POST 或 GET。当单击 Button 这样的控件时，会导致向服务器发送一个回发。在这个回发期间，表单中的所有相关信息都会被提交回服务器，在服务器上可以用来重构页面。默认情况下，所有的



ASP.NET Web 窗体总是使用 POST 方法向服务器发送数据。

在<form>表单中包含一个隐藏的_VIEWSTATE 字段，可以在下面的代码中找到：

```
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTYzNjg0OTc2OA9kFgICAw9kFgICBQ88KwAKAQAPFgIeAlNEFgEGAMAGys8EzghkZGR
cHifoNAWzLcinnOO8VhqGgqlonRdjna1EYDFzODVYhQ==" />
```

当加载 ASP.NET 页面时，ASP.NET 运行库会用关于该页面的信息填充这个隐藏字段。例如，当单击【设置时间】按钮引起一个回发时，它会为 Label1 控件的 Text 属性添加值。类似地，它还包含 Calendar 的选中日期。当通过回发提交回该页面时，就会通过请求发送该隐藏字段_VIEWSTATE 中的值。然后，当 ASP.NET 在服务器上创建了新页面时，_VIEWSTATE 字段中的信息就会被读取并应用到页面中的控件上。通过这种方式，像 Label 这样的控件就能维持它的文本。

当将 EnableViewState 属性设置为 False 时，也就关闭了 Label 控件的 ViewState。关掉这个设置后，ASP.NET 运行库不会再跟踪 Label 控件。因此，当单击【提交】按钮时，ASP.NET 运行库无法在 ViewState 中找到 Label 控件的任何信息，所以，标签显示它的默认文本“Label”。



3.4.2　如何关闭视图状态

并不是所有控件都一直依赖于 ViewState。有很多控件能维持它们自己的某些状态。这些控件包括 TextBox、CheckBox、RadioButton 和 DropDownList。它们能维持自身的值，这是因为它们在浏览器中被呈现为标准的 HTML 表单控件。例如，TextBox 服务器控件在客户端浏览器中得到的 HTML 代码如下所示：

```
<input name="TextBox1" type="text" value="Text" id="TextBox1" />
```

当发送回一个带有这样的 TextBox 的页面时，浏览器也会将控件的值发送回服务器。然后 ASP.NET 运行库就能够再次用这个值来预先填写文本框，而不需要从 View State 中获取值。显然，这也比将值存储在 ViewState 中更有效。如果存储在 ViewState 中，值就会被发送到服务器中两次：一次是在文本框中，另一次是在 ViewState 中。当值比较大时，ViewState 引擎便会大大增加页面的大小，增加页面加载的时间。因此，在不需要时最好关闭它。这样就能最小化隐藏字段_VIEWSTATE 的大小。

关闭 View State 很容易，可以在以下 3 个地方做到：

⦿ 在 Web 站点级别

可以在根站点的 Web.config 文件中通过修改<system.web>下面的<pages>元素，将 enableViewState 属性设置为 false 来完成。

```
<pages enableViewState="false">
  ...
</pages>
```

通常不在站点级别关闭 ViewState，因为在站点级别关闭 ViewState 后，将无法为特定的控件打开这个功能。幸运的是，ASP.NET 4.0 提供了一个新的属性 ViewStateMode，它提供了关于 ViewState 如何使用的更多控制。

- 在页面级别

在每个页面的页面指令中，可以将 EnableViewState 设置为 False，例如：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default"
EnableViewState="False" %>
```

对于确信根本不需要 View State 的页面来说，这种方法是非常有用的。

- 在控件级别

各个 ASP.NET 服务器控件允许分别设置 EnableViewState 属性，这样可以选择关闭某些控件，而使其他控件保持打开。

一旦在更高级别(web.config 或页面级别)上关闭了 ViewState，就不能再在一个低层级别(页面或特定控件级别)打开这个功能。但是，使用新的 ViewStateMode 属性仍能完成如下工作：

(1) 禁止在 web.config 文件中关闭 View State。

(2) 在页面级别，将 EnableViewState 设置为 True，将 ViewStateMode 设置为 Disabled，如下所示：

```
<%@ Page Language="C#"...EnableViewState="True" ViewStateMode="Disabled" %>
```

上述代码可以关闭页面中所有控件的 ViewState，除了那些再次明确地将 ViewStateMode 属性设置为 Enabled 的控件以外。

(3) 如果想让控件支持 ViewState，可以将控件的 ViewStateMode 设置为 Enabled，例如：

```
<asp:Label ID="Label1" runat="server" Text="Label" ViewStateMode="Enabled" />
```

读者可以尝试修改【例 3-12】，在 Default.aspx 的页面指令中设置 EnableViewState 为 True、将 ViewStateMode 设置为 Disabled。然后在页面中添加另一个 Label 控件，并将第一个 Label 控件的 ViewStateMode 设置为 Enabled：

```
<asp:Label ID="Label1" runat="server" Text="Label" ViewStateMode="Enabled" />
<asp:Label ID="Label2" runat="server" Text="Label" />
```

在按钮的单击事件处理程序中，将最新的日期和时间也赋予第二个标签控件：

```
Label1.Text = DateTime.Now.ToString();
Label2.Text = DateTime.Now.ToString();
```



重新运行程序，查看两个控件的变化情况。

3.5　上机练习

本章的上机练习主要介绍 FileUpload 控件和 AdRotator 控件的使用，ASP.NET 服务器控件的用法都比较相似，通过本章的上机练习希望读者能触类旁通，自己摸索其他控件的用法。

3.5.1　上传文件

上传文件是 Web 应用中比较常见的功能，在 ASP.NET 中，使用 FileUpload 控件可以快速开发实现文件的上传。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 3-1】。

(2) 切换到 Default.aspx 页面的设计视图，添加一个 FileUpload 控件、一个 Button 控件和一个 Label 控件到<div>标记中。控件的属性设置和布局如图 3-50 所示。

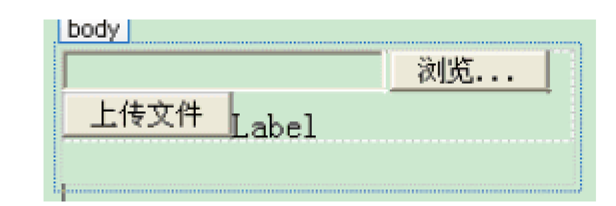


图 3-50　控件布局

(3) 双击 Button 控件，添加按钮的单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        bool bOK = false;
        string path = Server.MapPath("~/");
        if (FileUpload1.HasFile)
        {
            string fileExt = System.IO.Path.GetExtension(FileUpload1.FileName).ToLower();
            string[] allowExt = new string[] {".gif",".png",".jpeg",".jpg" };
            foreach (string str in allowExt)
                if (str == fileExt)
                    bOK = true;
        }
        if (bOK)
        {
            try
            {
                FileUpload1.PostedFile.SaveAs(path + FileUpload1.FileName);
```

```
                Label1.Text = "文件 " + FileUpload1.FileName + " 已成功上传";
            }
            catch (Exception ex)
            {
                Label1.Text = "文件上传失败 " + ex.Message;
            }
        }
        else
            Label1.Text = "只能上传.gif、.png、.jpeg、.jpg 类型的文件";
    }
}
```

(4) 编译并运行程序，在默认浏览器中加载 Default.aspx 页面，执行结果如图 3-51 所示。

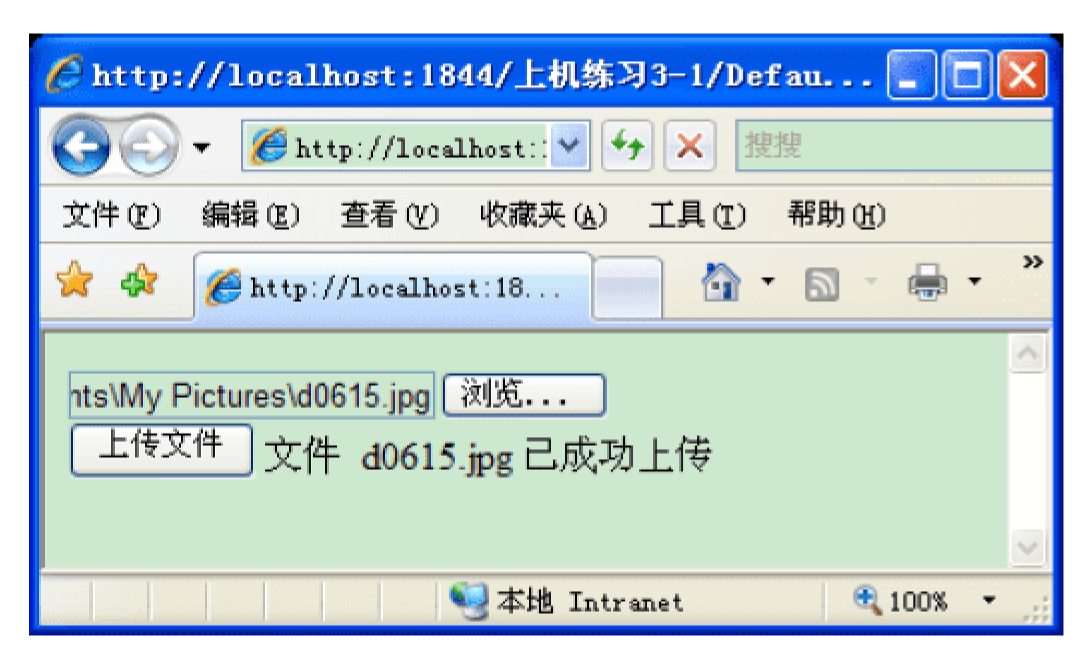


图 3-51 文件上传成功

3.5.2 使用 AdRotator 控件

AdRotator 控件提供了一种在 ASP.NET 页面中显示广告的方法，该控件可显示.gif 文件或其他图形文件，当用户单击广告时，系统会导航到指定的目标 URL。

AdRotator 控件通常与 XML 文件或数据库表配合使用，以显示指定的图片广告。本节的上机练习将使用 AdRotator 服务器控件显示 XML 数据文件中的广告。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 3-2】。

(2) 在网站根目录下新建名为 Images 的文件夹，将准备好的广告图片复制到该文件夹中。

(3) 在【解决方案资源管理器】面板中右击 App_Data 文件夹，从弹出的快捷菜单中选择【添加新项】命令，打开【添加新项】对话框，如图 3-52 所示，可以看到，这个【添加新项】对话框中可选的模板文件都是与数据文件相关的，在此我们选择【XML 文件】选项。

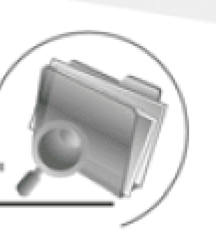




图 3-52　【添加新项】对话框



提示

建议将广告文件放置在 App_Data 文件夹中，因为 ASP.NET 可防止浏览器利用该文件夹中的文件。



(4) 向广告信息文件中添加下列如下 XML 元素：

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements xmlns="http://schemas.microsoft.com/AspNet/AdRotator-Schedule-File">
<Ad>
  <ImageUrl>~/images/tsinghua.jpg</ImageUrl>
  <NavigateUrl>http://www.tsinghua.edu.cn</NavigateUrl>
  <AlternateText>清华大学</AlternateText>
  <Impressions>100</Impressions>
</Ad>
<Ad>
  <ImageUrl>~/images/pku.gif</ImageUrl>
  <NavigateUrl>http://www.pku.edu.cn</NavigateUrl>
  <AlternateText>北京大学</AlternateText>
  <Impressions>50</Impressions>
</Ad>
<Ad>
  <ImageUrl>~/images/zju.jpg</ImageUrl>
  <NavigateUrl>http://www.zju.edu.cn</NavigateUrl>
  <AlternateText>浙江大学</AlternateText>
  <Impressions>100</Impressions>
</Ad>
```

```
<Ad>
  <ImageUrl>~/images/nju.jpg</ImageUrl>
  <NavigateUrl>http://www.nju.edu.cn</NavigateUrl>
  <AlternateText>南京大学</AlternateText>
  <Impressions>50</Impressions>
</Ad>
</Advertisements>
```

(5) 切换到 Default.aspx 的设计视图，在网页上要显示广告的位置添加一个 AdRotator 控件，单击控件右上角的小三角，弹出【AdRotator 任务】面板，单击【选择数据源】下拉列表，选择【<新建数据源>】选项，如图 3-53 所示。

(6) 在打开的【数据源配置向导】对话框中选择【XML 文件】数据源，单击【确定】按钮，如图 3-54 所示。

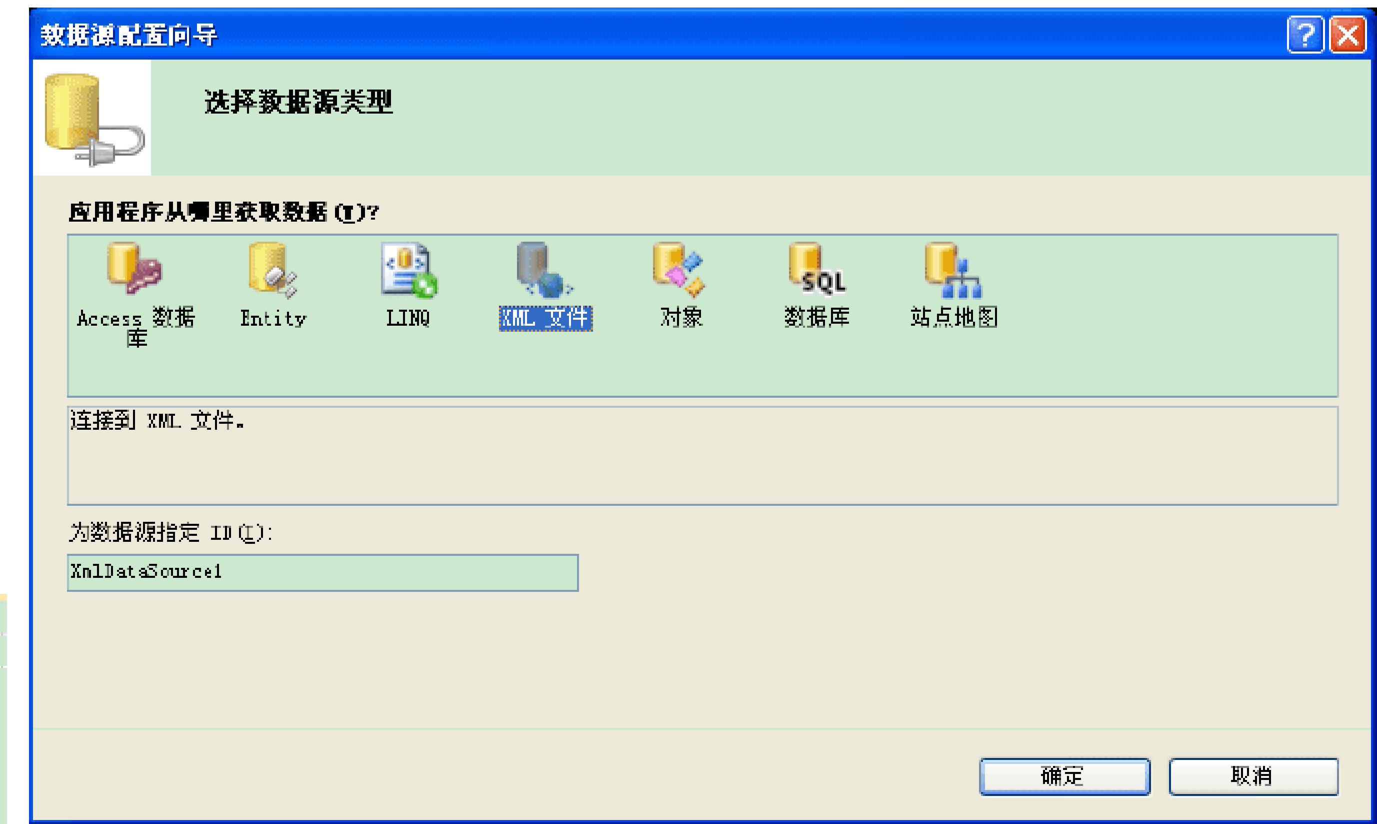


图 3-53 【AdRotator 任务】面板　　图 3-54 【数据源配置向导】对话框

(7) 在【配置数据源】对话框中，单击【数据文件】后面的【浏览】按钮，选择刚才创建的 XML 文件，如图 3-55 所示。

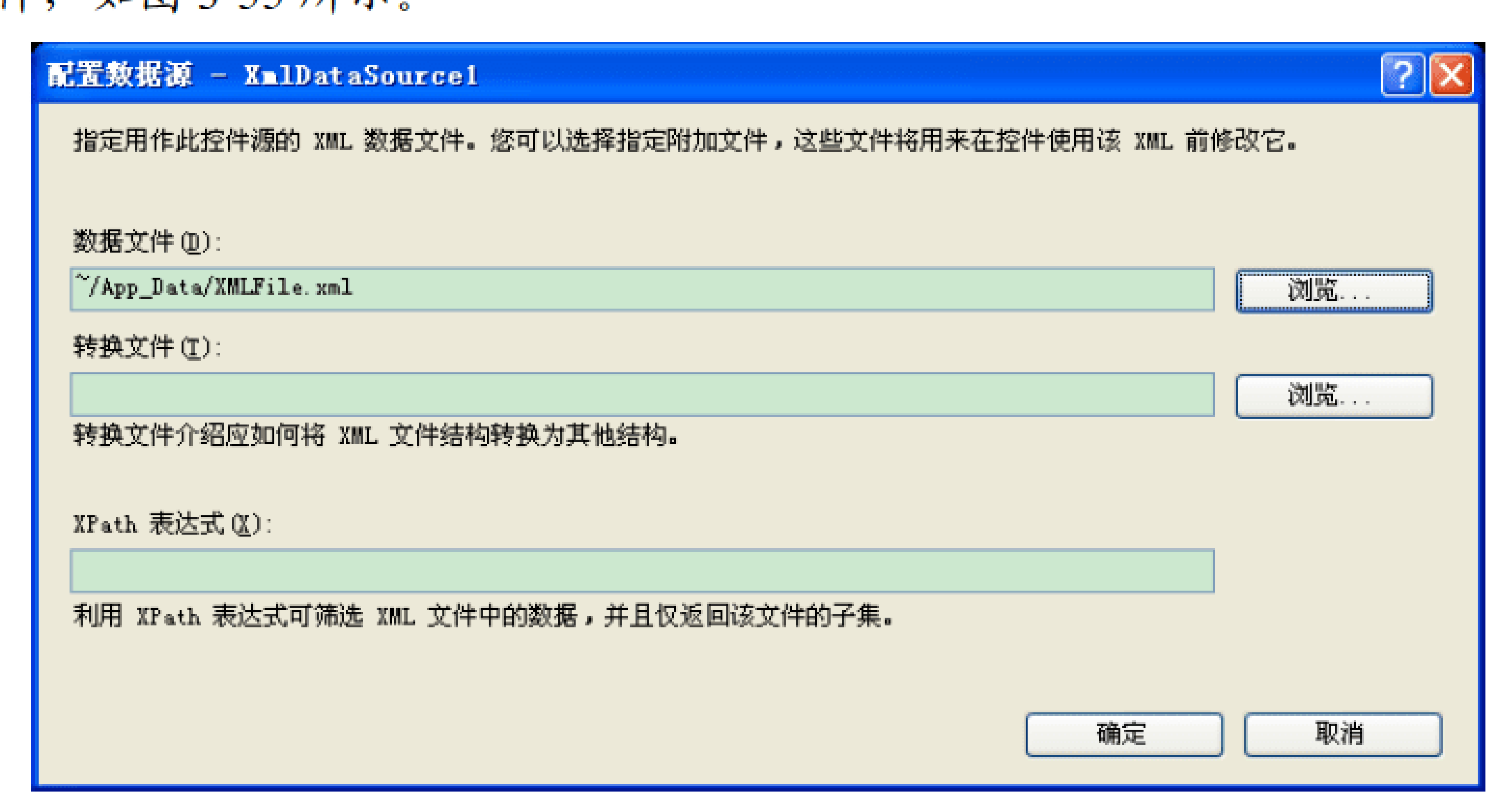
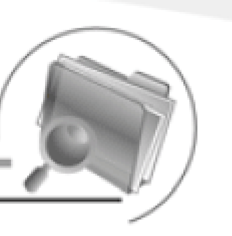


图 3-55 【配置数据源】对话框



(8) 编译并运行程序，在浏览器中打开 Default.aspx 页面，按【F5】键或者单击工具栏的【刷新】按钮可以显示不同的图片。

本例中创建的 XML 文件中有一个<Advertisements>节点，其中可以包括多个<Ad>节点，每个<Ad>对应一个广告图片。AdRotator 控件的所有属性都是可选的，<Ad>中可以包括下列属性：

- ImageUrl：要显示的图像的 URL。
- NavigateUrl：单击 AdRotator 控件时要跳转到的网页 URL。
- AlternateText：图像不可用时显示的文本。
- Keyword：可用于筛选特定广告的广告类别。
- Impressions：一个指示广告的可能显示频率的数值(加权数值)。在 XML 文件中，所以 Impressions 值的总和不能超过 2,048,000,000-1。
- Height：广告的高度，以像素为单位。
- Width：广告的宽度，以像素为单位。

3.6　习题



1. 如果将 RadioButton 控件进行分组？

2. ASP.NET 提供了几个验证控件，各有什么作用？

3. 当使用一个 CustomValidator 控件时，可以在客户端和服务器上编写有效性验证代码。如何告知 ASP.NET 运行库在有效性验证处理期间调用什么客户端有效性验证方法？

4. 使用 TreeView 控件有两种方式：一种方式是作为带项和子项的列表，单击它们时能折叠或展开；另一种方式是作为显示所有项的静态列表，不能折叠或展开。要禁止用户展开或折叠树中的项，需要设置控件上的什么属性呢？

5. ASP.NET 运行库如何在回发之间跟踪控件状态？

第4章 样式、主题和母版页

学习目标

开发 Web 应用程序通常需要考虑两个方面：功能和外观。ASP.NET 提供了一些可在应用程序中对页面、控件的外观和样式进行自定义的功能，例如可以为某个控件设置字体、背景色和前景色、宽度以及高度等样式。本章将全面介绍 Web 应用程序中样式控制和页面布局所用到的技术和使用方法，包括 CSS 样式、主题和母版页。这些技术对于创建具有一致外观的网站非常有用，也将使站点看起来更专业和更具吸引力。

本章重点

- 编写和应用 CSS 样式
- VWD 提供的大量快速编写 CSS 的工具
- 创建和应用主题
- 在主题中定义外观
- skinID 属性的使用
- 创建母版页和内容页

4.1 CSS 样式

Internet 出现伊始，Web 页面主要由文本和图像组成。文本是使用纯 HTML 格式化的，这种格式化所提供的样式化页面的选项很有限，因此诞生了 CSS 来弥补这方面的缺陷。

本节将介绍 CSS 的概念、在 VWD 2010 中如何使用 CSS 以及 CSS 的样式规则和用法。



4.1.1　HTML 格式化的缺点

使用 HTML 进行格式化的问题之一是它提供的样式化页面的选项很有限。可以用<em>、<strong>及<font>这样的标记来改变文本的外观，用 bgcolor 这样的属性来改变 HTML 元素的背景颜色，还有几个属性可用来改变链接出现在页面中的方式。显然，这个功能集不足以创建符合用户期望与需求的生动 Web 页面。

另一方面，在设计时，HTML 会强制要求在 HTML 文档中嵌入格式化信息，使得以后难以对设计进行重用或修改。

除了维护性问题之外，HTML 格式化的另一个问题是：在用户的浏览器中不能轻松地在运行时修改格式。

HTML 格式化的最后一个问题是，页面中的附加标记大大增加了页面的大小。这样，由于需要从 Web 站点中的各个页面上下载信息，下载和显示就会变慢。而且，当需要滚动大型的 HTML 文件来查找需要的内容时，页面也会变得难以维护。

简言之，使用 HTML 格式化存在如下问题：

- 它的有限功能集远远满足不了页面的格式化需求。
- 数据与表现混合在相同的文件中。
- HTML 无法在浏览器中于运行时轻松地切换格式。
- 必需的格式化标记与属性使页面更大，因此加载和显示更慢。

4.1.2　什么是 CSS

CSS(Cascading Style Sheet)，中文译为层叠样式表，是用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。就语法而言，CSS 是一种容易学习的语言。它的“语法”仅由几个概念组成，使得用户相当容易入门。

使用 CSS 样式可以非常灵活并更好地控制网页外观，大大减轻了用户实现精确布局定位、维护特定字体和样式的工作量。

CSS 规定了两种定义样式的方法，分别是内联式和级联式。

1. 内联式样式

直接将样式控制放在单个 HTML 元素内，称为内联式或行内样式。该样式通过 style 属性来控制每个元素的外观，这种方法直观但是很繁琐，除非具有相同样式的元素较少，否则很少采用。下面是一段采用内联式来控制各个元素外观的 CSS 示例代码：

```
<body style="text-align:center">
<form id="form1" runat="server">
<div style="text-align:center; width:400px; border:solid 1px blue">
```

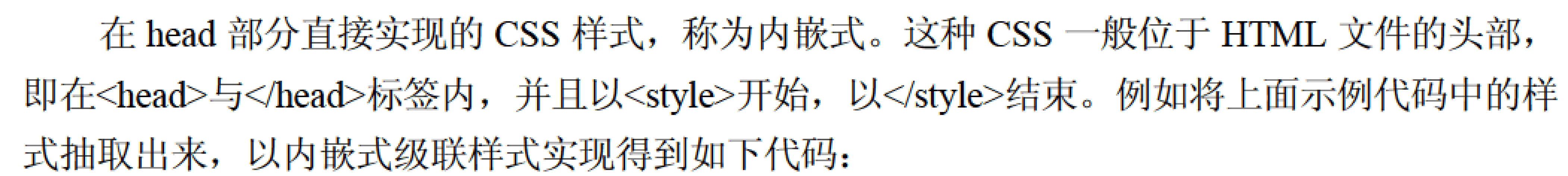
```
<h1 style="font-size:x-large; color:red ">欢迎光临</h1>
<h2 style="font-size:large; color:blue ">这是一个被 style 修饰的页面</h2>
</div>
</form>
</body>
```

2. 级联式样式

在网页的 head 部分定义或导入的样式，称为级联式样式。该样式可以实现将网页结构和表现分离，这样，当修改某些元素的样式时，只需要修改 head 部分定义或引入的样式，该网页内所有具有该样式的元素都会自动应用新的样式。

级联式样式又可以使用两种方式来控制样式：内嵌式和链接式。

◉ 内嵌式


在 head 部分直接实现的 CSS 样式，称为内嵌式。这种 CSS 一般位于 HTML 文件的头部，即在<head>与</head>标签内，并且以<style>开始，以</style>结束。例如将上面示例代码中的样式抽取出来，以内嵌式级联样式实现得到如下代码：

```
<head>
<title>内嵌式样式</title>
<style Type="text/css">
<!--
body{ text-align:center }
div{ text-align:center; width:400px; border:solid 1px blue }
h1{ font-size:x-large; color:red}
 h2{ font-size:large;   color:blue }
-->
</style>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1>欢迎光临</h1>
<h2>这是一个被 style 修饰的页面</h2>
</div>
</form>
</body>
```

其中<style>与</style>之间是样式的内容，在{ }前面可以写样式的类型和名称。{ }中是样式的属性。这种方法是经常被使用的添加样式表的方法。

可见，采用内嵌式比内联式方便了很多，body 内的代码也相对简洁，修改某个元素的样式时



只需修改 head 内的代码即可。

⊙　链接式

内嵌式只解决了一个网页内部结构和表现分离的问题，一般情况下网站都由很多网页组成，不同网页中的某些元素采用了相同的样式，仍然需要分别设置，因此，将样式放在一个单独的 CSS 文件中，然后通过为每个网页引入该文件来实现统一的外观将是一种更好的选择。

在 head 部分通过导入以扩展名为.css 的文件来实现 CSS 样式，称为链接式。利用这种方法在网页中可以调用已经定义好的样式表来实现样式表的应用。定义好的样式表通常单独以文件的形式存放在站点目录中。这种方法实现了将网页结构和表现的彻底分离，最适合大型网站的 CSS 样式定义。

例如将上面示例中的样式抽取出来以文件的形式存放，保存到一个名为 style.css 的文件中，内容如下：

```
body
{
    text-align:center;
}
div
{
    text-align:center;
    width:400px;
    border:solid 1px blue;
}
h1
{
    font-size:x-large;
    color:red;
}
h2
{
    font-size:large;
    color:blue;
}
```



在页面中，可以通过<link>标记引用样式 style.css 文件，代码如下所示：

```
<head>
    <title>链接式样式</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
```

```
        <form id="form1" runat="server">
            <div>
                <h1>欢迎光临</h1>
                <h2>这是一个被 style 修饰的页面</h2>
            </div>
        </form>
</body>
```



知识点

在引用样式的标记<link>中，ref 属性规定了 XHTML 与被链接文件的关系，href 属性指定了要链接的样式表文件的 URL，type 属性则规定了链接文件的类型。上述样式文件与当前页面存放在同一个目录下，如果不存放在同一个目录下，相应的<link href=" "...>标记中 href 属性的值要有所改变。



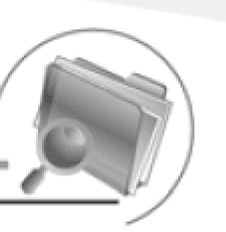
⦿ 样式嵌套

此外，如果某个元素既引用了链接样式文件中定义的样式，又在 head 部分定义了新的样式，或者在元素内部通过 style 属性定义了新的样式，那么该标记元素最终呈现的效果会是什么样呢？下面通过一个例子来说明这个问题。

【例 4-1】样式嵌套举例。

```
<head>
<title>链接式样式</title>
<style Type="text/css">
<!--
h1{ font-weight:bold }
    h2{ color: yellow}
-->
</style>
<link rel="stylesheet" href="style.css" type="text/css">
</head>
<body>
<form id="form1" runat="server">
<div>
<h1 style=" font-size:small ">欢迎光临</h1>
<h2 style=" font-weight:bold ">这是一个被 style 修饰的页面</h2>
</div>
</form>
</body>
```

其中，style.css 文件是前面创建的样式文件。运行这个 HMTL 文件，在浏览器中可以看到，



h1 元素内的文字以粗体、小号、红色显示，而 h2 元素内的文字则以粗体、大号、蓝色显示。可见，链接式样式中 h1 元素的 font-size 属性和内嵌式样式中 h2 元素的 color 属性都没有起作用，而不冲突的样式则都会起作用。这就是样式嵌套中的冲突问题，浏览器解决这种问题的方法就是一旦发现样式冲突，则通过“就近使用”原则，采用距离该元素最近的样式进行显示，而不冲突的样式则通过顺序组合后形成最终样式进行显示。



知识点

设计者可以根据实际情况选择一种或多种样式控制方法进行样式定义。一般情况下，在样式表(.css)文件中定义适合大多数网页公用的样式，在网页内部采用内嵌式定义该页面特有的样式，内联式样式定义个别元素的样式，再结合可视化的开发工具，从而使样式控制真正灵活、方便。

4.1.3　CSS 属性简介



属性是元素的一部分，可通过样式表修改。CSS 规范定义了一个长属性列表，但在大多数 Web 站点中不会用到所有项。如表 4-1 所示列出了部分常见的 CSS 属性及其应用场合。

表 4-1　常见的 CSS 属性

CSS 属性	描　　述	示　　例
background-color background-image	指定元素的背景色或图像	background-color:White; background-image: url(Image.jpg);
border	指定元素的边框	border: 3px solid black;
color	修改字体颜色	color: Green;
display	修改元素的显示方式，允许隐藏或显示它们	display: none; 这种设置使元素被隐藏，不占用任何屏幕空间
float	允许用左浮动或右浮动将元素浮动在页面上。其他内容则被放在相应的位置上	float: left; 该设定使跟着一个浮动的其他内容被放在元素的右上角。在本章后面将介绍它的工作原理
font-family font-size font-style font-weight	修改页面上使用的字体外观	font-family: Arial; font-size: 18px; font-style: italic; font-weight: bold;
height width	设置页面中元素的高度或宽度	height: 100px; width: 200px;

(续表)

CSS 属性	描　述	示　例
margin padding	设置元素内部(内边距)或外部(页边距)的可用空间	padding: 0; margin: 20px;
visibility	控制页面中的元素是否可见。不可见的元素仍然会占用屏幕空间，只是看不到它们而已	visibility: hidden; 这会使元素不可见。但仍然会占用页面的原始空间

VWD 会通过它的许多 CSS 工具帮助用户找到恰当的属性，因此不必全部记住它们。

4.2　在 VWD 中使用 CSS



VWD 中有以下几个使用 CSS 的便利工具。

- 【样式表】工具栏：用来快速访问并创建新规则与样式。
- 【CSS 属性】面板：用来修改属性值。
- 【管理样式】窗口：用来组织站点的样式，将它们从内嵌样式表改为外部样式表，反之亦然；对它们重新排序；将现有样式表链接到一个文档；创建新的内联、内嵌或外部样式表。
- 【应用样式】窗口：用来从站点中选择所有可用样式，并将它们快速地应用到页面中的不同元素上。
- 【添加样式规则】对话框：用于构建较复杂的选择器。

4.2.1　创建新样式

在 VWD 2010 中使用 CSS 非常方便，即可以在源视图中利用 VWD 的智能感知功能设置各种样式，也可以利用可视化的对话框和便利工具快速完成各种样式的设置。

1. 在源视图下设置样式

在源视图下，利用系统提供的智能提示功能，可以方便地设置各种元素的内联式样式，具体步骤如下：

(1) 在要设置格式的 HTML 标记内，输入 style="，然后按空格键，将弹出 VWD 2010 提供的智能感知工具，如图 4-1 所示。

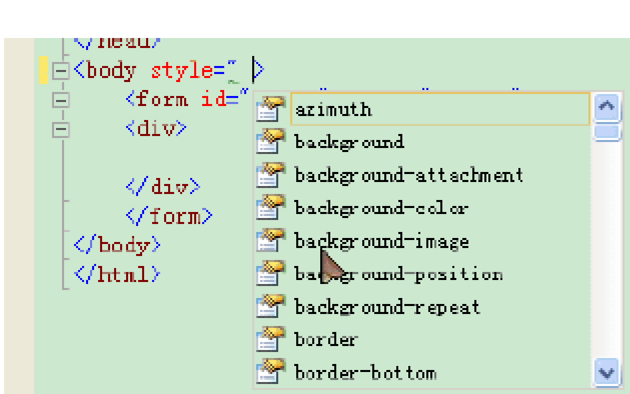


图 4-1　VWD 的智能感知工具



(2) 定义任意数量的属性(“属性:值”对)，属性之间用分号分隔。

2. 在可视化窗口中设置样式

利用可视化窗口设置样式的方法有很多，可以在源视图或者设计视图下选中某个标记元素，然后单击【属性】面板中 style 属性后面的省略号按钮(…)，将打开【修改样式】对话框，如图 4-2 所示。

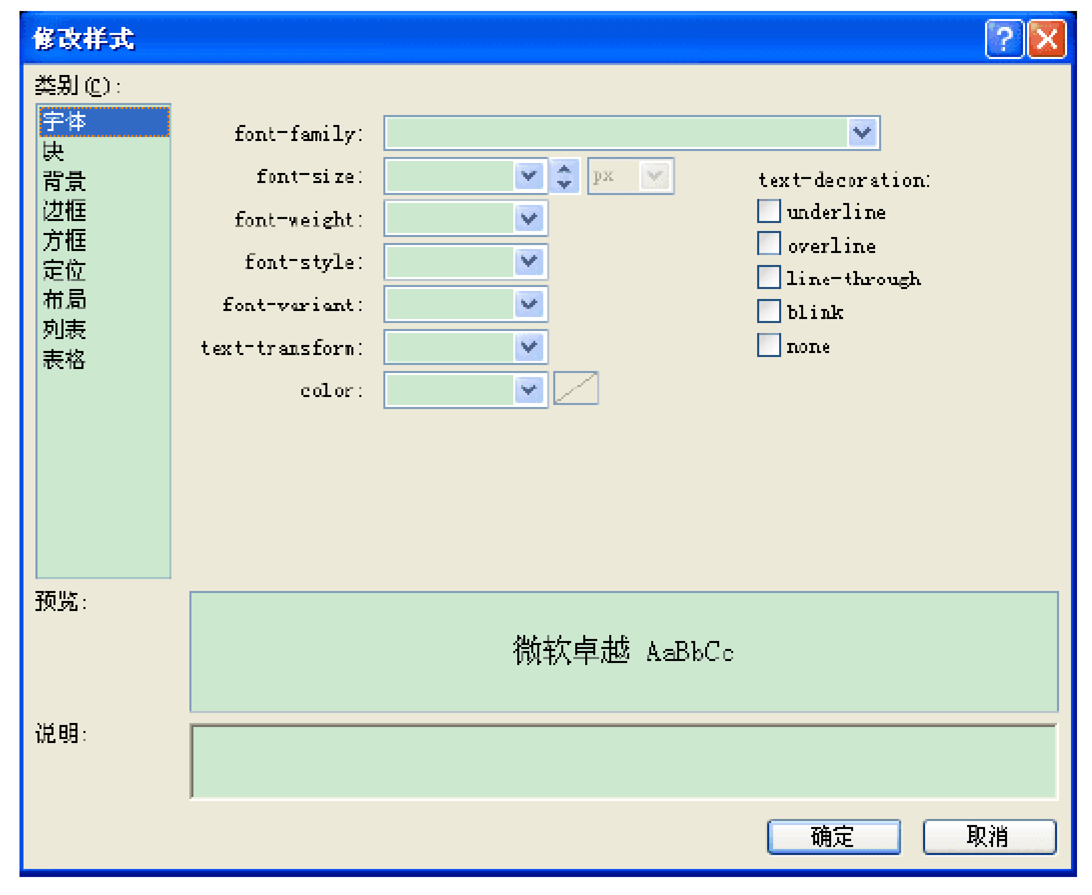


图 4-2　【修改样式】对话框



该对话框分为两个窗格，左窗格列出 9 个类别，当选择某个类别时，右窗格显示所选类别下的选项。设置了样式选项并单击【确定】按钮后，新的样式定义将自动在源视图中生成，也可以在设计视图下查看最新的效果。

这种方法只能将定义的样式属性以内联式生成，放在每个元素的 style 属性中。要想定义内嵌式样式也非常容易，具体步骤如下：

(1) 切换到设计视图，在【格式设置】工具栏的【目标规则】列表中，选择【应用新样式】选项，如图 4-3 所示。



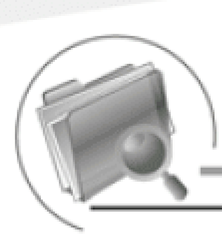
图 4-3　选择应用新样式

(2) 此时将打开【新建样式】对话框，该对话框与【修改样式】对话框相似，所不同的是【新建样式】对话框中包含了【选择器】，用于选择对哪一个标记进行定义，以及通过【定义位置】将当前定义存放到哪里，如图 4-4 所示。



知识点

选择【格式】|【新建样式】命令，也可以打开【新建样式】对话框。



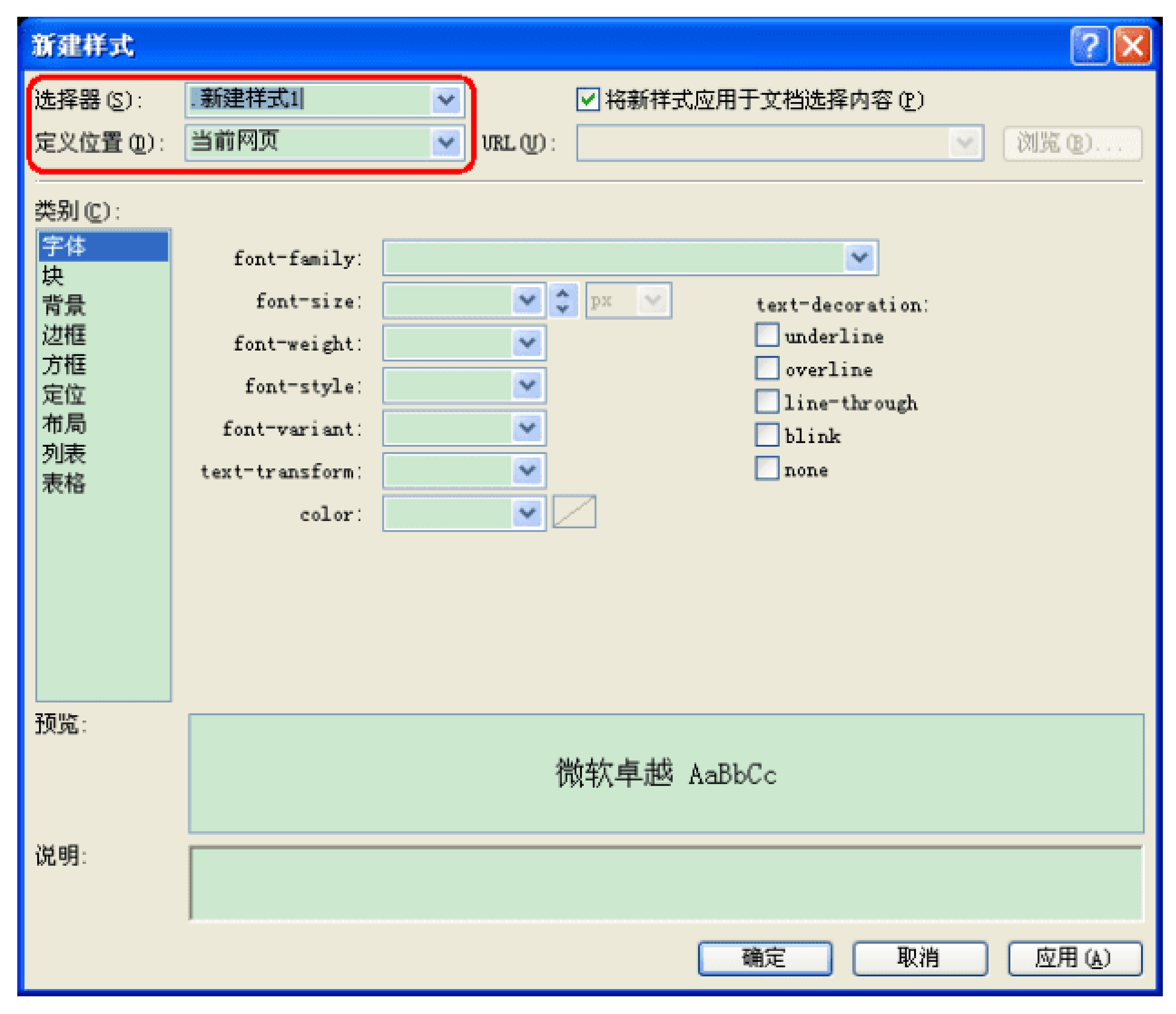


图 4-4 【新建样式】对话框

在【选择器】列表中选择某个选择器，如 h1，就可以创建应用于所有 h1 元素的样式。

【定义范围】列表设置为【当前网页】，表示该样式规则在当前页的 style 元素中创建。若想查看已创建的样式规则，可以切换到源视图并滚动到 style 元素，该元素位于<head>标记内。

3. 使用【CSS 属性】面板

可以使用【CSS 属性】面板对正在定义的内嵌式样式规则进行修改，具体步骤如下：

(1) 单击要修改样式的元素，选中部分以蓝色框包围并有一个标签指示 h3 元素已选中。

(2) 选择【视图】|【其他窗口】|【CSS 属性】命令，打开【CSS 属性】面板，可以看到 h3 元素的 CSS 属性列表，如图 4-5 所示。

(3) 通过该面板可以根据需要设置 CSS 属性。

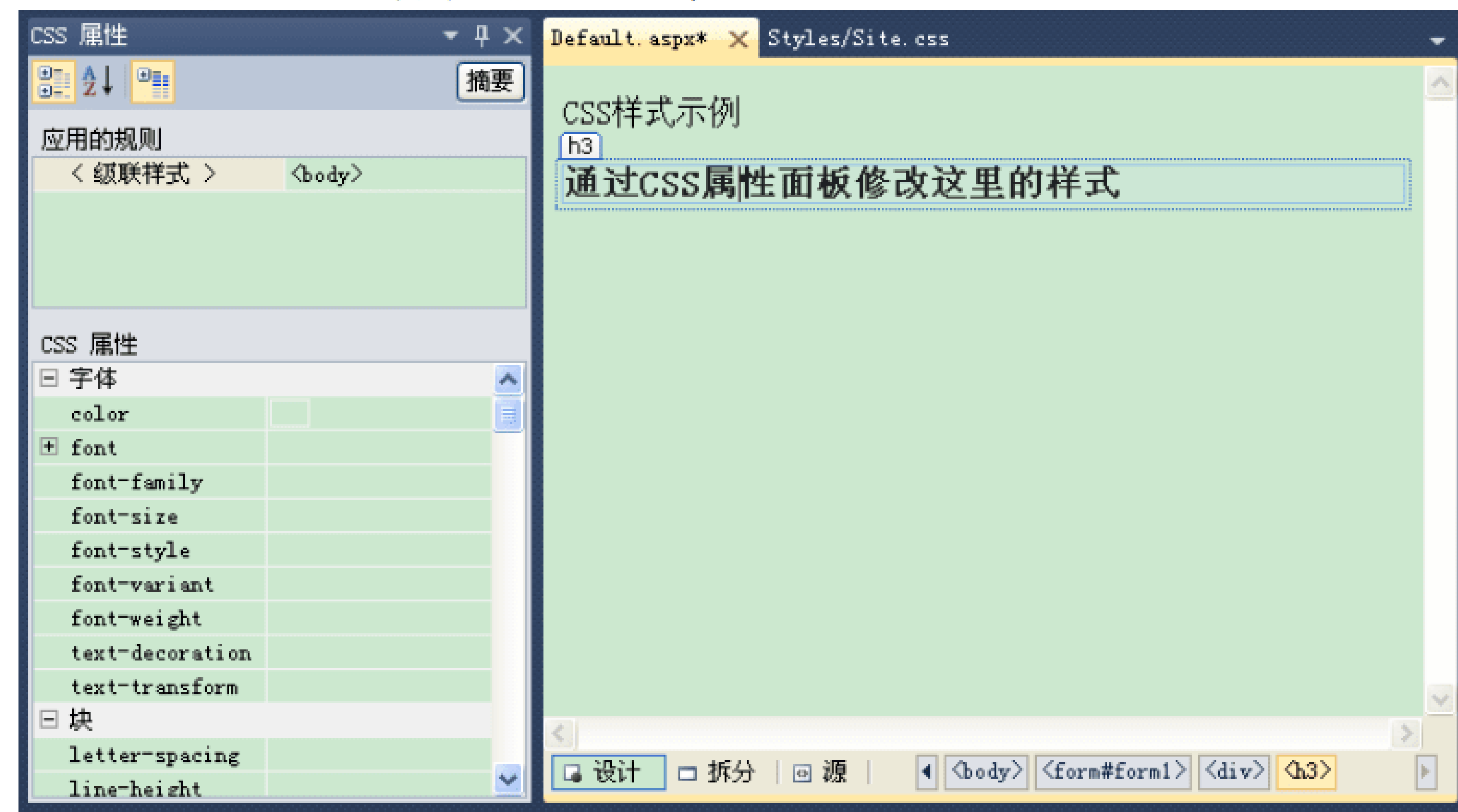
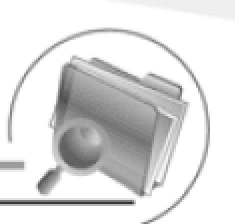


图 4-5 【CSS 属性】面板





在 CSS 属性列表中所做的修改可以立即通过设计视图显示出来，如果要观察样式的代码，可以切换到源视图并滚动到 style 元素处进行查看。

4. 新建样式表

使用 CSS 的另一个有效方法是将样式规则放入样式表中，然后所有页面都可以引用这些样式，这样可以使这些页面看起来非常一致。创建样式表的具体步骤如下：

(1) 在【解决方案资源管理器】中，右击解决方案的名称，从弹出的快捷菜单中选择【添加新项】命令。打开【添加新项】对话框，在【模板】列表中选择【样式表】选项，在【名称】文本框中输入样式表的名称 StyleSheet.CSS，单击【添加】按钮。

(2) 此时，编辑器将打开一个包含空 body 样式规则的新样式表。

(3) 打开或切换到 Default.aspx 页，切换到设计视图。选择【格式】|【附加样式表】命令，打开【选择样式表】对话框，选择刚才创建的样式表文件 StyleSheet.css，单击【确定】按钮，如图 4-6 所示。



图 4-6　【选择样式表】对话框

(4) 执行上述操作后，在 Default.aspx 的源视图中可以看到，在<head>标记中将添加了如下代码：

```
<link href="StyleSheet.css" rel="stylesheet" type="text/css" />
```



知识点

可以通过多种方式为页面指定样式表。最简单的方法是在源视图中将文件从【解决方案资源管理器】中拖到页面的<head>标记中或者直接将文件拖到设计视图中。

4.2.2　样式规则

一个样式表由若干个样式规则组成。样式规则是指网页元素的样式定义，包括元素的显示方

式以及元素在页中的位置等。打开前面添加的样式表文件 StyleSheet.css，在其中右击鼠标，在弹出的快捷菜单中选择【添加样式规则】命令，如图 4-7 所示，打开如图 4-8 所示的【添加样式规则】对话框。

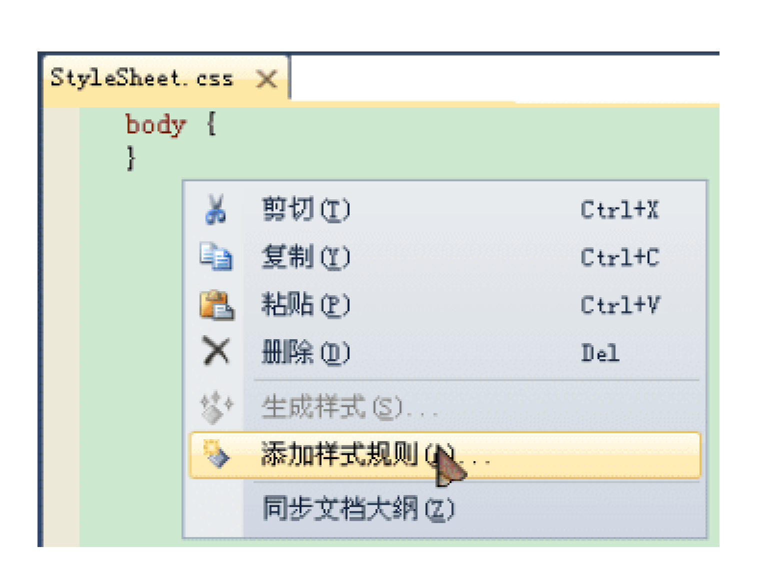



图 4-7　选择【添加样式规则】命令

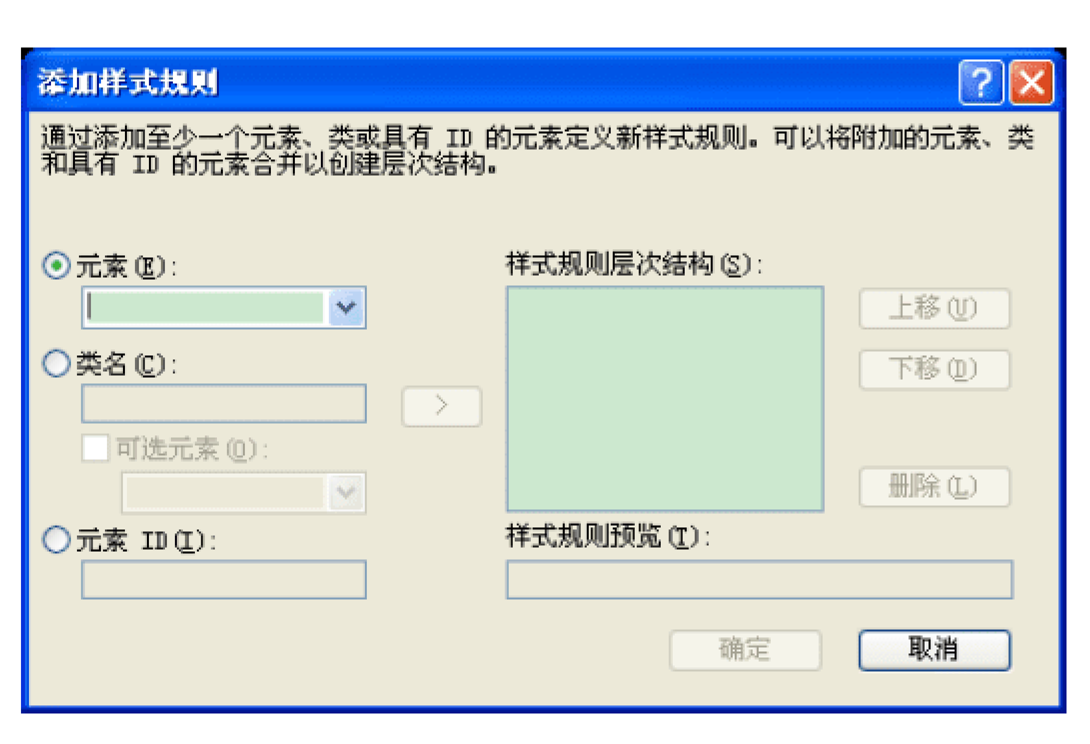


图 4-8　【添加样式规则】对话框

在【添加样式规则】对话框中选择某个元素，或者定义一个类，或者定义一个元素 ID，单击【确定】按钮即可添加一个新的样式规则。例如，添加一个元素 h1，在样式表文件中可以看到如下新建的样式规则：

```
h1
{
}
```

该规则默认是仅有元素名称的空规则，在大括号内右击鼠标，从弹出的快捷菜单中选择【生成样式】命令，即可打开【修改样式】对话框。

无论是定义内嵌式样式还是链接式样式，每个样式的定义格式都是一样的，如下所示：

```
样式定义选择符{ 属性 1:值 1; 属性 2:值 2; …… }
```

其中，样式定义选择符是指样式定义的对象，可以是 HTML 标记元素，还可以是用户自定义的类、ID、伪类和伪元素等。

1. 标记选择符

任何 HTML 元素都可以是一个 CSS 的标记选择符。标记选择符仅仅是指向特别样式的元素。【添加样式规则】对话框中的【元素】下拉列表中提供了所有可供使用的标记选择符。

2. 类选择符

每一个标记选择符都能自定义不同的类，从而允许同一元素具有不同的样式。指定某个标记选择符内的自定义类的一般形式为：

```
标记选择符.类名{样式属性 1:值 1; 样式属性 2:值 2; ……}
例如：
```



```
p.one{
color:red;
}
p.two{
color:blue;
}
```

在代码中引用类选择符的方法是通过元素的 class 属性来实现的。代码如下：

```
<p class="one">类别选择器 1</p>
<p class="two">类别选择器 2</p>
```

其含义是在 p 中引用 one 会以红色样式显示，在 p 中引用 two 会以蓝色样式显示。

在【添加样式规则】对话框中先选中【类名】单选按钮，在文本框中输入 one，然后选中【可选元素】复选框，在其下拉列表中选择 p 元素，即可自动生成 p.one 样式规则，如图 4-9 所示。

类选择符的定义也可以与标记选择符无关，这样，类选择符可以应用于任何元素。这种自定义类选择符的形式如下：

```
.类名{样式属性 1:值 1; 样式属性 2:值 2; ……}
```

创建这种类选择符时，只要不选中【可选元素】复选框即可，如图 4-10 所示。

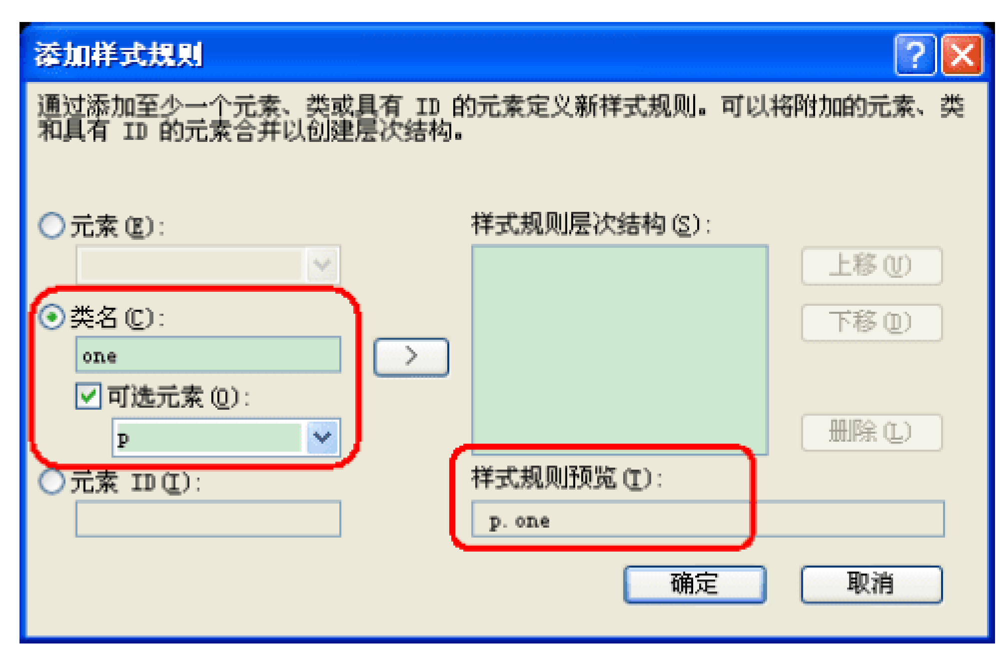


图 4-9　添加 p.one 样式规则

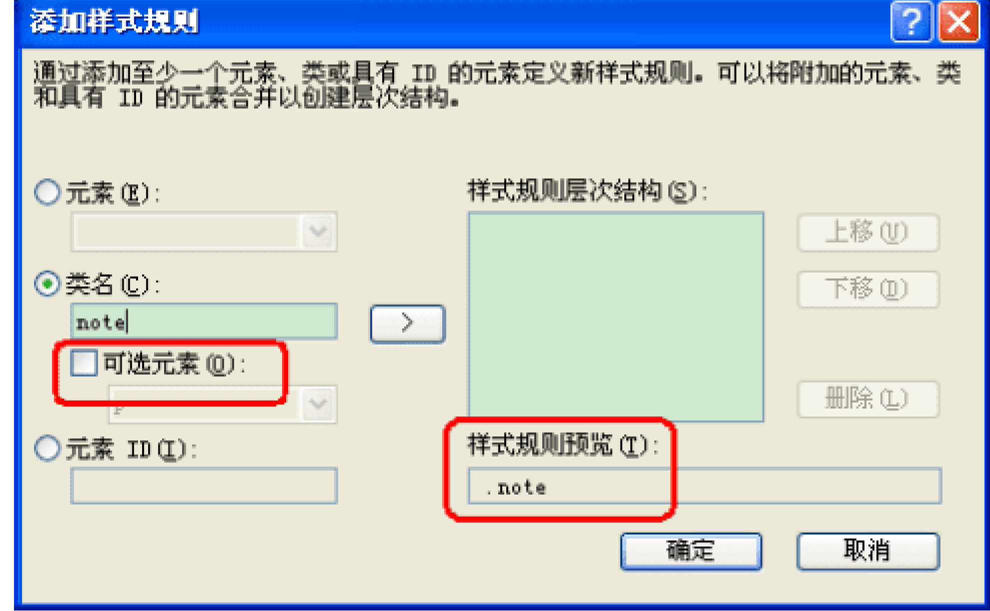


图 4-10　添加.note 样式规则

例如：

```
<style type="text/css">
.note{
color:red;
}
</style>
<h1 class="note">类别选择器 1</p>
<h2 class="note">类别选择器 2</p>
```

在这个例子，note 类选择符可被用于任何元素。



3. ID 选择符

ID 选择符用于分别定义每个具体元素的样式。一个 ID 选择符的指定要有指示符#在名字前面。使用时通过指定元素的 id 属性来关联。例如：

```
#index { color:blue }
```

引用时，使用 id 属性声明即可。

```
<p id="index">本段落的颜色为蓝色</p>
```

自定义 ID 选择符与自定义类选择符的定义方式非常相似，在【添加样式规则】对话框中，选中【元素 ID】单选按钮，输入相应的名称即可。但是两者在使用上是有区别的：在同一个网页中，多个标记元素可以使用同一个自定义类选择符，而 ID 选择符只能为某一个标记元素使用。这种选择符应该尽量少用，因为它有一定的局限。




提示

如果在一个元素的样式定义中，既有标记选择符，又有自定义类选择符和自定义 ID 选择符，那么自定义 ID 选择符的优先级最高，其次是自定义类，标记选择符的优先级最低。

4. 伪类

伪类是 CSS 中非常特殊的类，能自动地被支持 CSS 的浏览器所识别。伪类可以指定 XHTML 中的 A 元素以不同的方式显示链接(links)、已访问链接(visited links)和可激活链接(active links)。其中，一个已访问链接可以定义为不同颜色的显示，甚至不同字体大小和风格。

CSS 中用 4 个伪类来定义链接的样式，分别是 a:link、a:visited、a:hover 和 a:active，例如：

```
a:link{font-weight : bold ;text-decoration : none ;color : #C00000 ;}
a:visited {font-weight : bold ;text-decoration : none ;color : #C30000 ;}
a:hover {font-weight : bold ;text-decoration : underline ;color : #F60000 ;}
a:active {font-weight : bold ;text-decoration : none ;color : #F90000 ;}
```

以上语句分别定义了“链接、已访问过的链接、鼠标停在上方时、点下鼠标时”的样式。注意，必须按以上顺序写，否则显示可能和预想的不一样。

5. 关联选择符

关联选择符是一个用空格隔开的两个或更多的单一标记选择符组成的字符串。一般格式如下：

```
选择符 1   选择符 2   ……  {属性:值; ……}
这些选择符具有层次关系，并且它们的优先级比单一的标记选择符大。例如：
p h1 { color:red }
```

这种定义方式只对 p 所包含的 h1 元素起作用，单独的 p 或者单独的 h1 元素均无法采用该样



式。在【添加样式规则】对话框中先添加 p 选择符，单击【>】按钮将其添加到【样式规则层次结构】中，然后再添加 h1 元素，如图 4-11 所示，如果层次结构有变化，还可以通过【上移】和【下移】按钮进行修改。

这种方式不仅适用于标记选择符，还可以关联自定义用户类，自定义 ID 以及任何样式选择符。

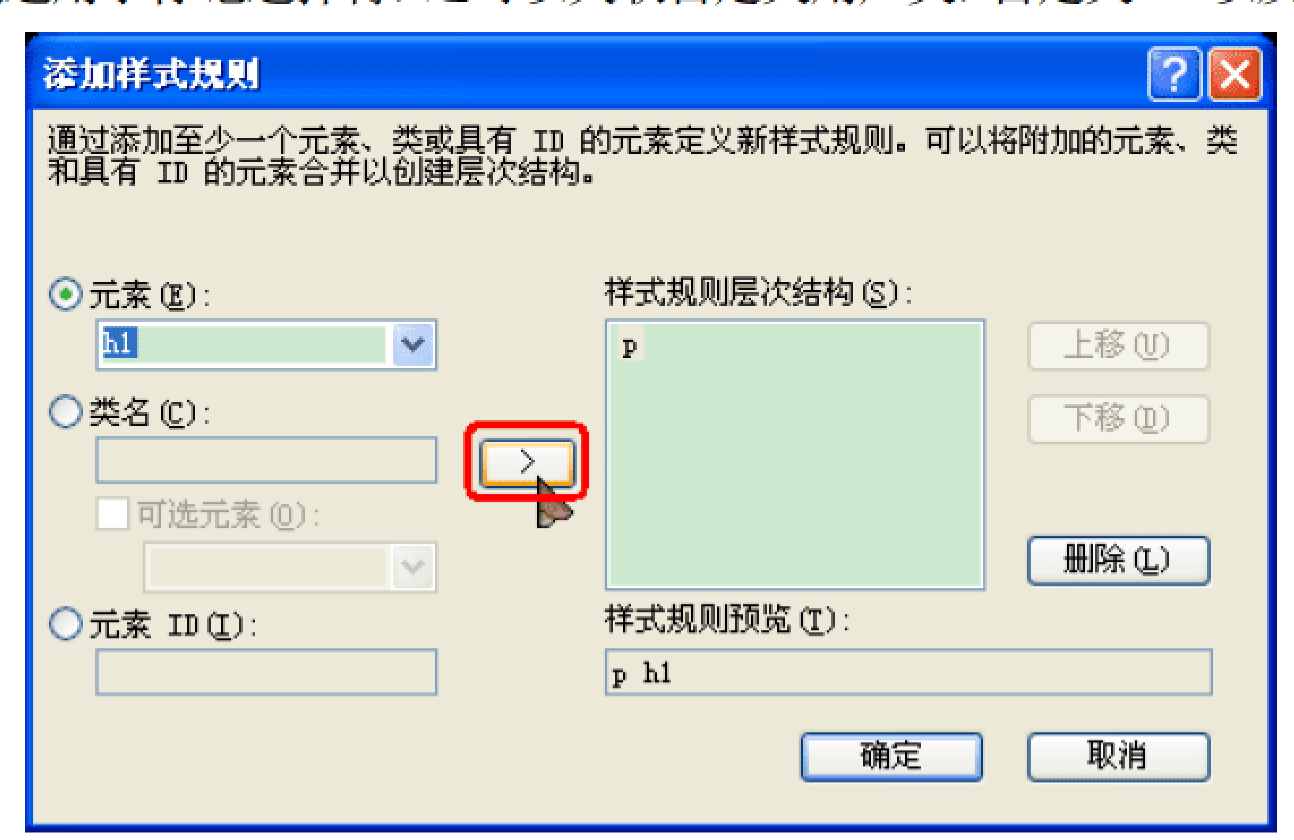


图 4-11　定义关联选择符

6. 并列选择符

如果由多个不同的元素定义的样式相同，则可以使用并列选择符简化定义。例如：

```
h1,h2,h3{ color:blue}
```

每个元素之间用逗号隔开，表示 h1、h2、h3 标记中的内容都将以蓝色样式显示。



知识点

直接在代码编辑器中输入规则很快捷。然而，在创建复杂的规则时，【添加样式规则】对话框可以帮助理解和创建规则的层次结构。【修改样式】对话框是创建新 CSS 声明的极佳工具。不需要记住各种 CSS 属性和它们的值，只要简单地在一个组织好的对话框中指定并一起单击即可。各种不同的 CSS 属性都根据逻辑被分组到不同类别下，以便更容易找到和修改它们。

4.2.3　应用样式

在 VWD 中，通过【应用样式】对话框可以快速地对页面中的元素应用已经定义好的 CSS 样式。选择【视图】|【其他窗口】|【应用样式】命令打开【应用样式】窗口，可轻松地向页面中的元素应用样式规则。

通过【应用样式】窗口对页面中的元素应用样式规则的具体操作如下：



(1) 首先已经创建好了样式表，并且将样式表附加到了页面。切换到页面的设计视图，选择【视图】|【其他窗口】|【应用样式】命令，打开【应用样式】窗口。

(2) 该窗口显示了它在当前页面中找到的所有选择器和附加的任何样式表。如果没有看到，可以单击该窗口工具栏中的【选项】按钮，选择【显示所有样式】命令。

知识点

为了帮助找到正确的样式，【应用样式】窗口中用红、绿、蓝和黄色点分别表示 ID 选择器、类选择器、元素选择器和内联样式。

(3) 将光标定位到要应用样式的元素处，如<p>元素，此时【应用样式】窗口中将显示当前可用的样式，单击 p.one 类，如图 4-12 所示。

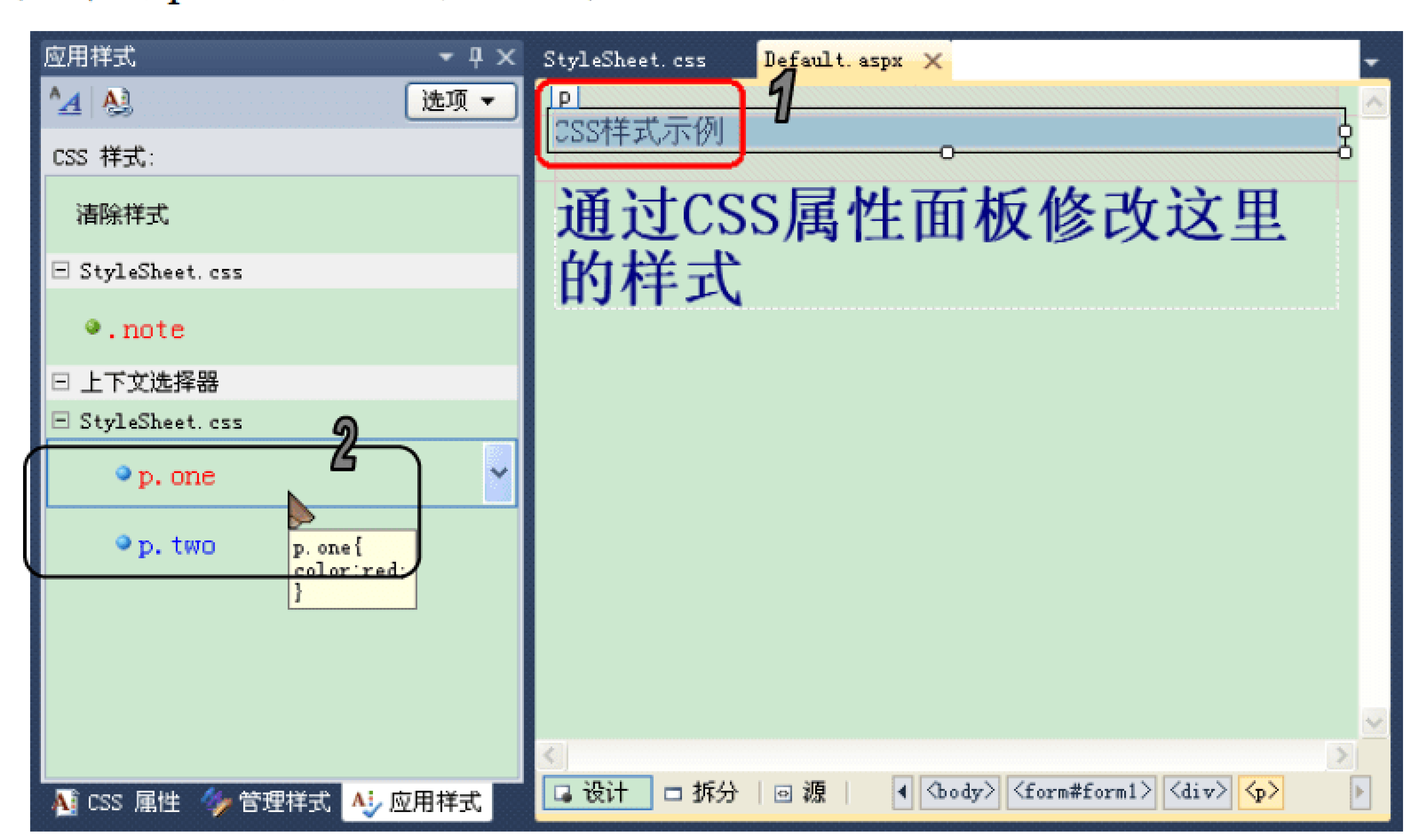


图 4-12　应用样式

(4) 切换到源视图，可以看到 VWD 会向<p>标记添加一个 class 属性：

```
<p class="one" >CSS 样式示例</p>
```

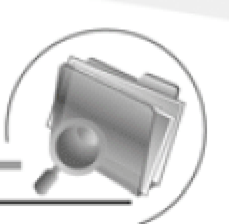
(5) 如果要应用多个类，则在单击列表中的其他类时，按住 Ctrl 键，这样就会应用一个类列表，元素的 class 属性之间由一个空格隔开。也可以对源视图中选中的元素应用同样的步骤。

(6) 单击【应用样式】窗口中的【清除样式】，就可以快速地从标记中删除现有类和内联样式。

4.2.4　管理样式

【管理样式】窗口提供了一个总体视图，可以看到应用到当前文档的所有外部和内嵌样式表。它是一个非常有用的窗口，可以将新样式附加到当前文档中，将一个位置的样式移到另一个位置。

下面的例 4-2 演示了【管理样式】窗口的应用。



【例 4-2】将内嵌式样式移到样式表文件中。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 4-2】。

(2) 切换到 Default.aspx 的设计视图，选择【格式】|【新建样式】命令，打开【新建样式】对话框。

(3) 在【选择器】下拉列表中选择 h2 标记，在【定义位置】下拉列表中选择“当前网页”，定义一个内嵌式样式 h2，切换到源视图，在<head>标记中将生成如下代码:

```
<style type="text/css">
    h2
    {
        color: #FF00FF;
        font-family:  宋体, Arial, Helvetica, sans-serif;
    }
</style>
```

(4) 使用前面学过的知识，新建一个 CSS 样式文件 StyleSheet.css，代码如下:

```
body
{
    font-family:  宋体, Arial, Helvetica, sans-serif;
}
p.one{
    color:red;
}
p.two{
    color:blue;
}
h1
{
    color: #000080;
}
```

(5) 将 StyleSheet.css 样式文件附加到 Default.aspx 页面。

(6) 选择【视图】|【其他窗口】|【管理样式】命令，打开【管理样式】窗口，此时该窗口中列出了样式文件中的样式和当前网页中的内嵌样式。

(7) 右击当前网页中的 h2 样式，从弹出的快捷菜单中选择【新建样式副本】命令，如图 4-13 所示。将打开【新建样式】对话框，允许创建一个基于当前选项的新样式，在【选择器】下拉列表中将默认自动选择 h2，在【定义位置】下拉列表中选择“现有样式表”选项，在【URL】下拉列表中选择 StyleSheet.css 选项，如图 4-14 所示。



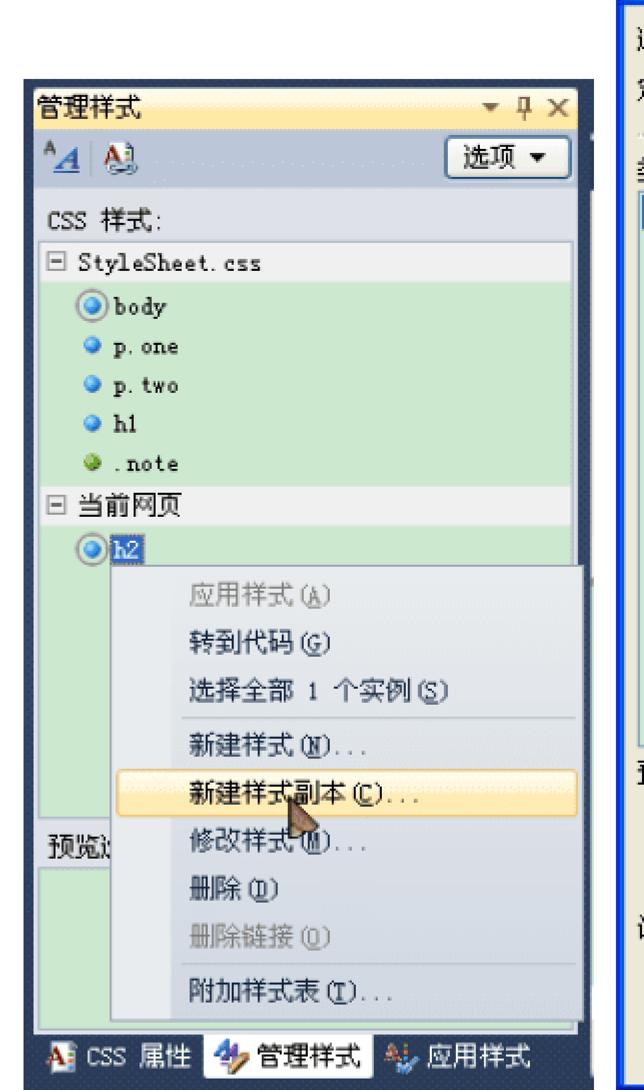


图 4-13 【管理样式】面板

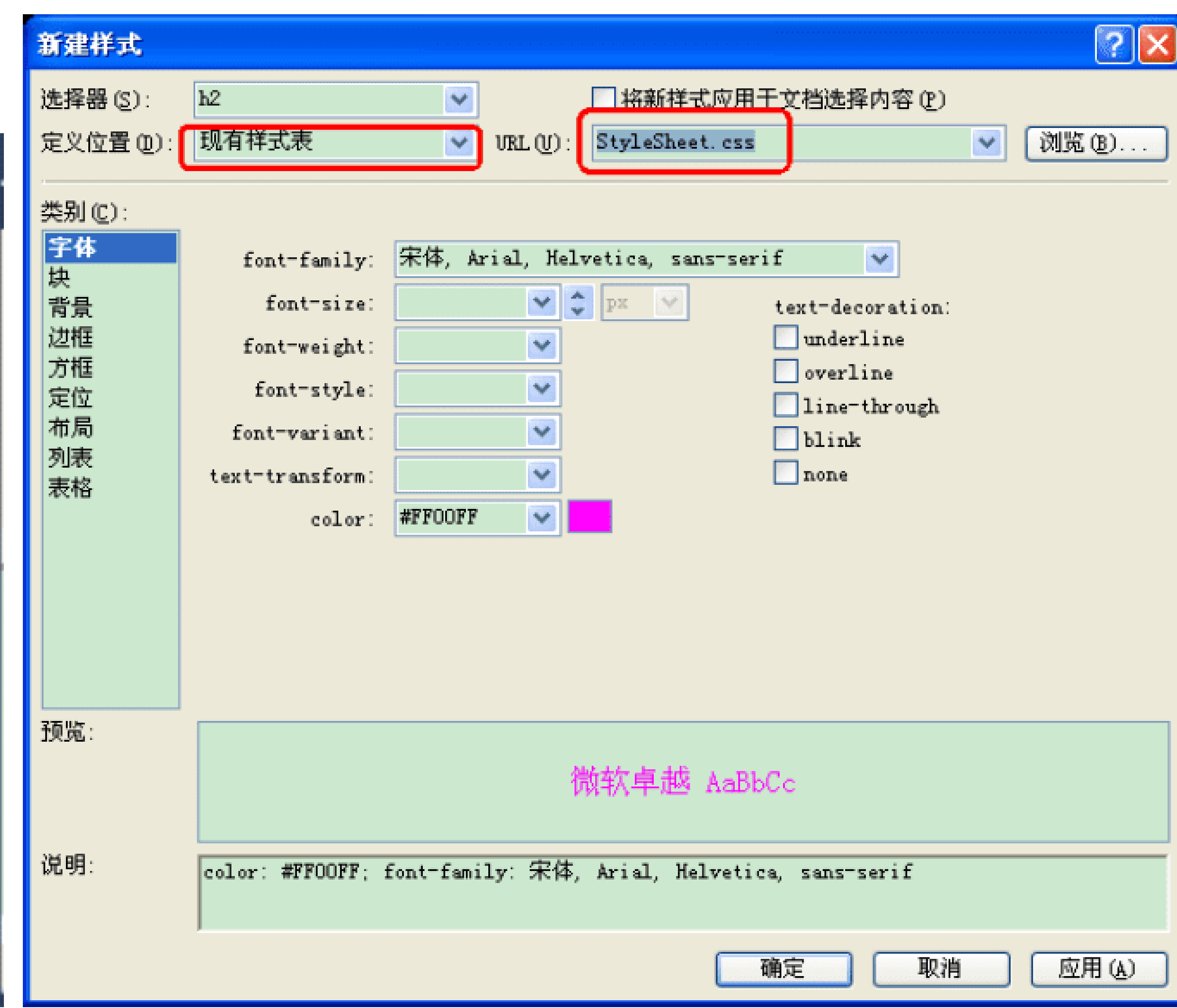


图 4-14 【新建样式】对话框

(8) 单击【确定】按钮关闭对话框。VWD 会创建 h2 样式的一个副本，并把它放在文件 StylesSheet.css 中。

(9) 在【管理样式】面板中，再次右击当前网页中的 h2 样式，从弹出的快捷菜单中选择【删除】命令，这样会把样式属性从<head>标记内删除。

通过以上操作完成了将内嵌式样式移到样式表文件中的操作。

4.2.5 DIV 和 CSS 布局

在传统的表格布局中，完全依赖于表格对象 TABLE，在页面中绘制多个单元格，在表格中放置内容，通过表格的间距或者用无色透明的 GIF 图片来控制布局版块的间距，达到排版的目的。而以 DIV 对象为核心的页面布局中，通过层来定位，通过 CSS 定义外观，最大程度地实现了结构和外观彻底分离的布局效果，因此习惯上对层布局又称为 DIV 和 CSS 布局。

层布局最核心的标签就是 DIV。DIV 是一个容器，在使用时以<DIV></DIV>形式存在。在 XHTML 中，每一个标签都可以称作是容器，能够放置内容。但 DIV 是 XHTML 中专门用于布局设计的容器对象。

1. 定义层

添加层的方法非常简单，可以通过【工具箱】面板中的 HTML 选项卡托拽一个 Div 项到设计视图中，或者在源视图中创建一对<div></div>标记。



2. 盒子模型

1996 年 CSS1 推出后，W3C 组织就建议把所有网页上的对象都放在一个盒子(box)中，设计师可以通过创建定义来控制这个盒子的属性，这些对象包括段落、列表、标题、图片以及层。盒子模型主要定义了 4 个区域：内容(content)、边框距(padding)、边界(border)和边距(margin)，如图 4-15 所示。

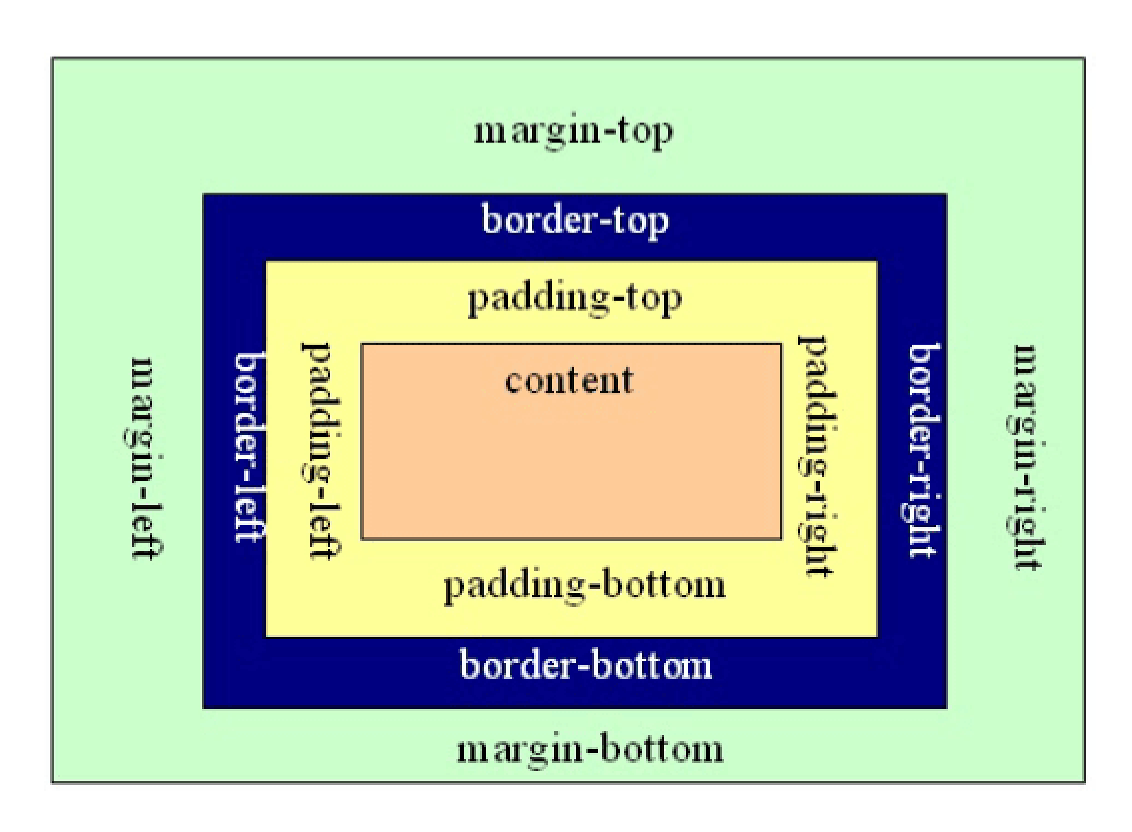


图 4-15　盒子模型



理解盒子模型就可以理解层与层之间定位的关系以及层内部的表达样式。其中，margin 属性负责层与层之间的距离，padding 属性负责内容和边框之间的距离。

下面的代码定义了盒子模型中的一些样式。

```
<style>
#sample2
{
background-color: #FFFF00;
border-style: solid;
padding-bottom: 25px;
margin-bottom: 50px;
width: 60%;
}
</style>
```

3. 层的定位

在一个页面中定义多个层，会发现这些层自动排列在不同的行，而要真正实现左右排列，就要加入新的属性——float(浮动属性)。float 浮动属性是 DIV 和 CSS 布局中的一个非常重要的属性。大部分的 DIV 布局都是通过 float 的控制来实现的。具体参数如下：

- float:none 用于设置是否浮动；
- float:left 用于表示对象向左浮动；

- float:right 用于表示对象向右浮动。

例如，下面的代码将创建一种左右上下分栏的样式，其效果如图 4-16 所示。

```
<head runat="server">
  <style>
   #left,#right{background-color:#eeeeee;border:1px solid #33ccff;height:200px; }
   #left{width:180px; float:left; }
   #bottom{ background-color:#eeeeee; border:1px solid #33ccff; height:50px; clear:both; }
  </style>
</head>
<body>
  <form id="form1" runat="server">
    <div id="left">当前层的 ID 是 left</div>
    <div id="right">当前层的 ID 是 right</div>
    <div id="bottom">当前层的 ID 是 bottom</div>
  </form>
</body>
```

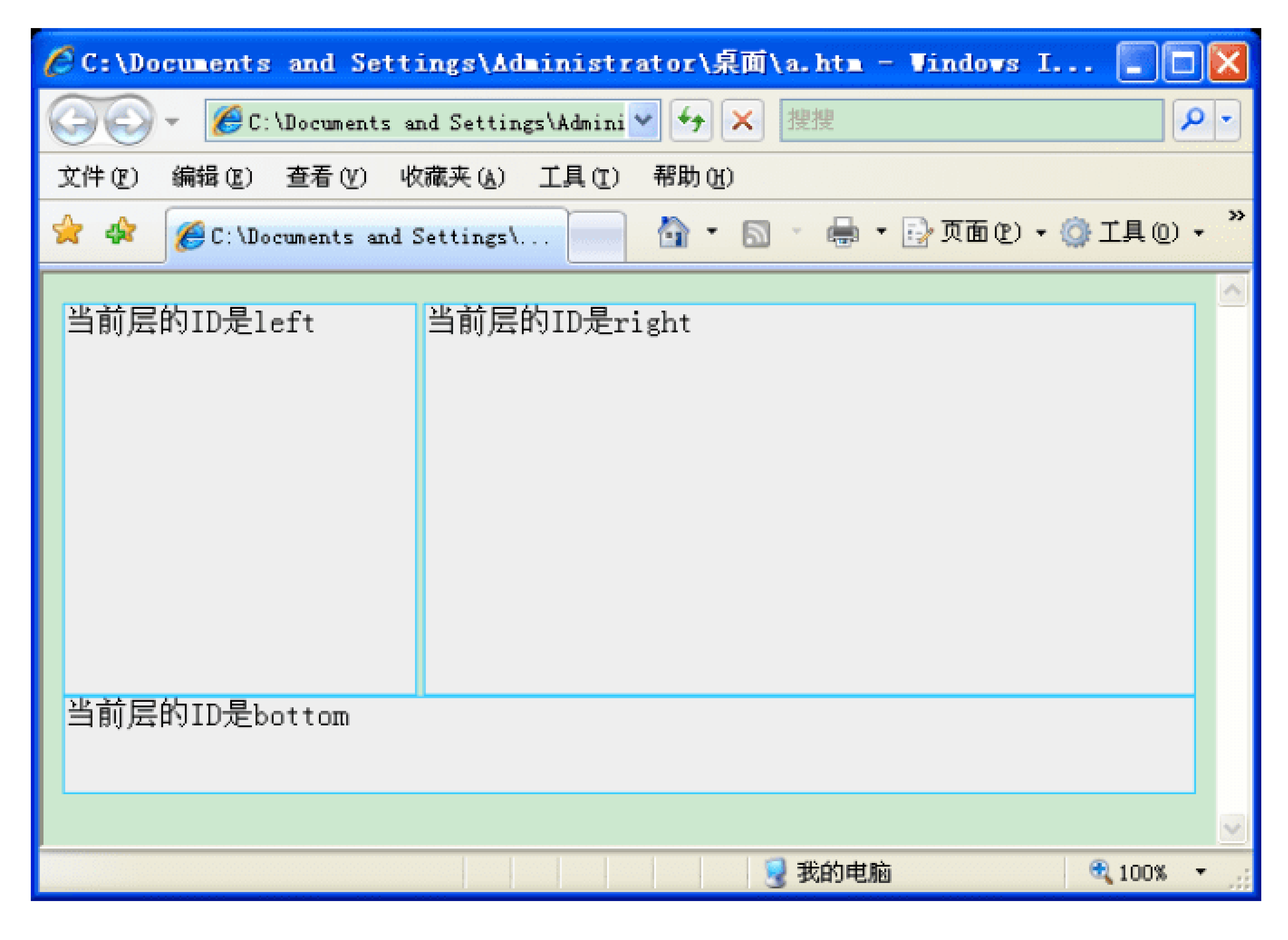
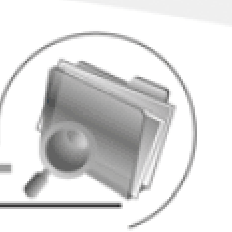


图 4-16　左右上下分栏

4. 利用 DIV 和 CSS 实现页面布局

DIV 只是一个区域标识，划定了一个区域，要实现样式还需要借助于 CSS，这样的分离，使得 DIV 的最终效果是由 CSS 来编写的。CSS 可以实现左右分栏，也可以实现上下分栏，而表格则没有这么大的灵活性。CSS 与 DIV 的无关性，决定了 DIV 在设计上有极大的伸缩性，而不拘泥于单元格固定的模式束缚。因此，实现网页布局，通常是先在网页中将内容用 DIV 标记出来，然后再用 CSS 来编写样式。



采用 DIV 和 CSS 布局之前，首先要分析网页有哪些内容块，以及每个内容块的含义，这就是所谓的网页结构。通常情况下页面结构包含以下几块。

(1) 标题区(header)：用来显示网站的标志和站点名称等。

(2) 导航区(navigation)：用来表示网页的结构关系，如站点导航，通常放置主菜单。

(3) 主功能区(content)：用来显示网站的主题内容，如商品展示、公司介绍等。

(4) 页脚(footer)：用来显示网站的版权和有关法律声明等。

通常采用 DIV 元素来将这些结构先定义出来，类似这样：

```
<div id="header"></div>
<div id="globalnav"></div>
<div id="content"></div>
<div id="footer"></div>
```

然后在 CSS 样式表中定义每个元素 ID 的具体样式，从而控制整个页面的布局。例如：

```
#content
{
    width: 740px;
    margin-top: 0px;
    margin-left:auto;
    margin-right:auto;
}
```



4.3　主题

网站的美观主要涉及页面和控件的样式属性，在 ASP.NET 应用程序中，可以利用 CSS 控制页面上各元素的样式以及部分服务器控件的样式，但是，有些服务器控件的属性无法通过 CSS 进行控制。为了解决这个问题，从 ASP.NET 2.0 开始就提供了一种称为“主题”的新方式，它可以保持网站外观的一致性和独立性，同时使页面的样式控制更加灵活方便，例如动态实现不同用户界面的切换等。

4.3.1　主题的基本概念

主题是指页面和控件外观属性设置的集合。主题由一个文件组构成，包括外观文件(扩展名为.skin)、级联样式表文件(扩展名为.css)、图片和其他资源等的组合，但一个主题至少包含一个外观文件。

1. 外观文件

外观文件是主题的核心文件，也称为皮肤文件，专门用于定义服务器控件的外观。在主题中可以包含一个或多个外观文件，外观文件的后缀名为.skin。

在控件外观设置中，只能包含主题的属性定义，如样式属性、模板属性、数据绑定表达式等，不能包含控件的 ID，如 Label 控件的外观设置代码如下：

```
<asp:Label runat="server" BackColor="Blue" Font-Names="Arial Narrow" />
```

这样一旦将该外观应用到 Web 页面中，则所有的 Label 控件都将显示外观所设置的样式。

右击某一个主题文件夹，从弹出的快捷菜单中选择【添加新项】命令，在打开的【添加新项】对话框中选择【外观文件】选项，并在【名称】文本框中输入外观文件名，单击【添加】按钮即可添加一个外观文件。

2. 级联样式表文件

主题中可以包含一个或多个 CSS 文件，一旦 CSS 文件被放在主题中，则应用时无需再在页面中指定 CSS 文件链接，而是通过设置页面或网站所使用的主题即可，当主题得到应用时，主题中的 CSS 文件会自动应用到页面中。

右击某一个主题文件夹，从弹出的快捷菜单中选择【添加新项】命令，在打开的【添加新项】对话框中选择【样式表文件】选项，即可添加样式表文件。

主题在 Web 站点的根文件夹的特殊文件夹 App_Themes 中。这个文件夹中可以包含多个主题目录，而外观文件等资源则放在主题目录中，如图 4-17 所示的主题目录结构中创建了 3 个主题，分别是“主题 1”、“主题 2”和“主题 3”，“主题 1”中包含一个外观文件，“主题 2”中包含两个外观文件，“主题 3”中包含一个外观文件和一个样式表文件。

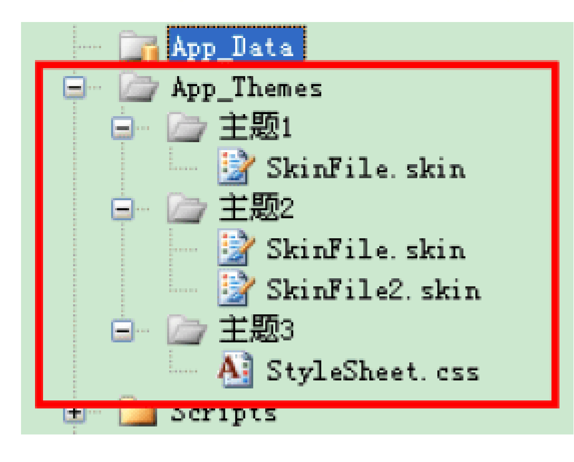


图 4-17　主题目录结构


知识点

每当主题处于活动状态时，对主题文件夹中各个 CSS 文件的链接就会自动添加到页面的<head>部分。

4.3.2　主题的类型

主题分为两大类型：一类是应用程序主题，另一类是全局主题。

- 应用程序主题是指保存在 Web 应用程序的 App_Themes 文件夹下的一个或多个主题文件夹，主题的名称就是文件夹的名称。
- 全局主题是指保存在服务器上，根据不同的服务器配置决定的，能够对服务器上所有



Web 应用程序起作用的主题文件夹。

一般情况下，很少用到全局主题，而本书所讲的主题仅指应用程序主题，即保存在应用程序中 App_Themes 文件夹下的主题文件夹，简称主题。

ASP.NET 页面有两个不同的设置主题的属性：Theme(页主题)属性和 StyleSheetTheme(页的样式表主题)属性。这两个属性都使用在 App_Themes 文件夹中定义的主题。虽然一开始它们看起来非常相似，但是在运行时它们的行为就不同了。StyleSheetThemes 在页面的生命周期中应用得非常早，在创建页面实例后不久就应用了。这意味着单个页面能通过在控件上应用内联属性来重写主题的设置。例如，带有将按钮的 BackColor 设置为紫色的外观文件的主题可以被页面标记中下面的控件声明重写：

```
<asp:Button ID="Button1" runat="server" Text="Button" BackColor="Blue" />
```

而 Theme 属性在页面的生命周期中应用的时间较晚，能有效地重写为单个控件自定义的任何属性。

由于 StyleSheetTheme 的属性能被页面重写，而 Theme 又能再次重写这些属性，两者用于不同的目的。如果想为控件提供默认设置则应设置 StyleSheetTheme，即 StyleSheetTheme 能为控件提供默认值，然后又可以在页面级重写。如果想强制应用控件的外观则应使用 Theme 属性。因为 Theme 中的设置不能再重写，而且它有效地重写了任何自定义设置，因此能确保控件的外观就是在主题中定义的样子。



如果由于某种原因不想向特定控件应用外观，可以通过设置控件的 EnableTheming 属性来禁用外观，如下：

```
<asp:Button ID="Button1" runat="server" EnableTheming="False" Text="Button" />
```

由于将 EnableTheming 属性设置为了 False，因此就不会向控件应用外观，而仍然会应用主题的 CSS 文件中的 CSS 设置。

4.3.3　创建并应用主题

打开一个 Web 应用程序，在【解决方案资源管理器】中，右击项目名，从弹出的快捷菜单中选择【添加】|【添加 ASP.NET 文件夹】|【主题】命令，系统自动创建 App_Themes 文件夹，并在该文件夹下生成一个默认名为“主题 1”的文件夹。在 App_Themes 文件夹中可以创建多个主题，方法相同。

为了创建一个主题，需要做下列事情：

- 如果站点中还没有 App_Themes 文件夹，则首先要创建该文件夹。
- 对于要创建的每个主题，用主题的名称创建一个子文件夹。
- 可选地，创建一个或多个将成为主题一部分的 CSS 文件。虽然根据主题命名 CSS 文件有助于标识正确的文件，但并不要求一定要这样做。添加到主题的文件夹中的任何 CSS

文件都会在运行时自动添加到页面中。

- 可选地，向主题文件夹中添加一个或多个图像。CSS 文件应当用稍后将介绍的相对路径来引用这些图像。
- 可选地，向主题文件夹中添加一个或多个外观文件。外观允许为之后要在运行时应用的特定控件定义单个属性(比如 ForeColor 和 BackColor)。

执行了这些步骤以后，就能将站点或单个 Web 页面配置为使用此主题。

1. 在主题中定义外观

skin 文件必须直接在主题的文件夹中创建。不能像存储主题的图像那样把 Skin 文件存储在一个子文件夹中。

在外观文件中，系统没有提供控件属性设置的智能提示功能，这使得用户定义自己的控件及其属性比较困难。要想使用 VWD 的智能提示功能，可做如下设置：

(1) 选择【工具】|【选项】命令，打开【选项】对话框。

(2) 展开【文本编辑器】选项，然后选择【文件扩展名】。

(3) 在右侧的【扩展名】文本框中输入 skin，然后从【编辑器】下拉列表中选择【用户控件编辑器】选项。

(4) 单击【添加】按钮，然后单击【确定】按钮完成设置，如图 4-18 所示。

设置完以后，当再次打开一个 skin 文件时，就会出现智能提示功能了。

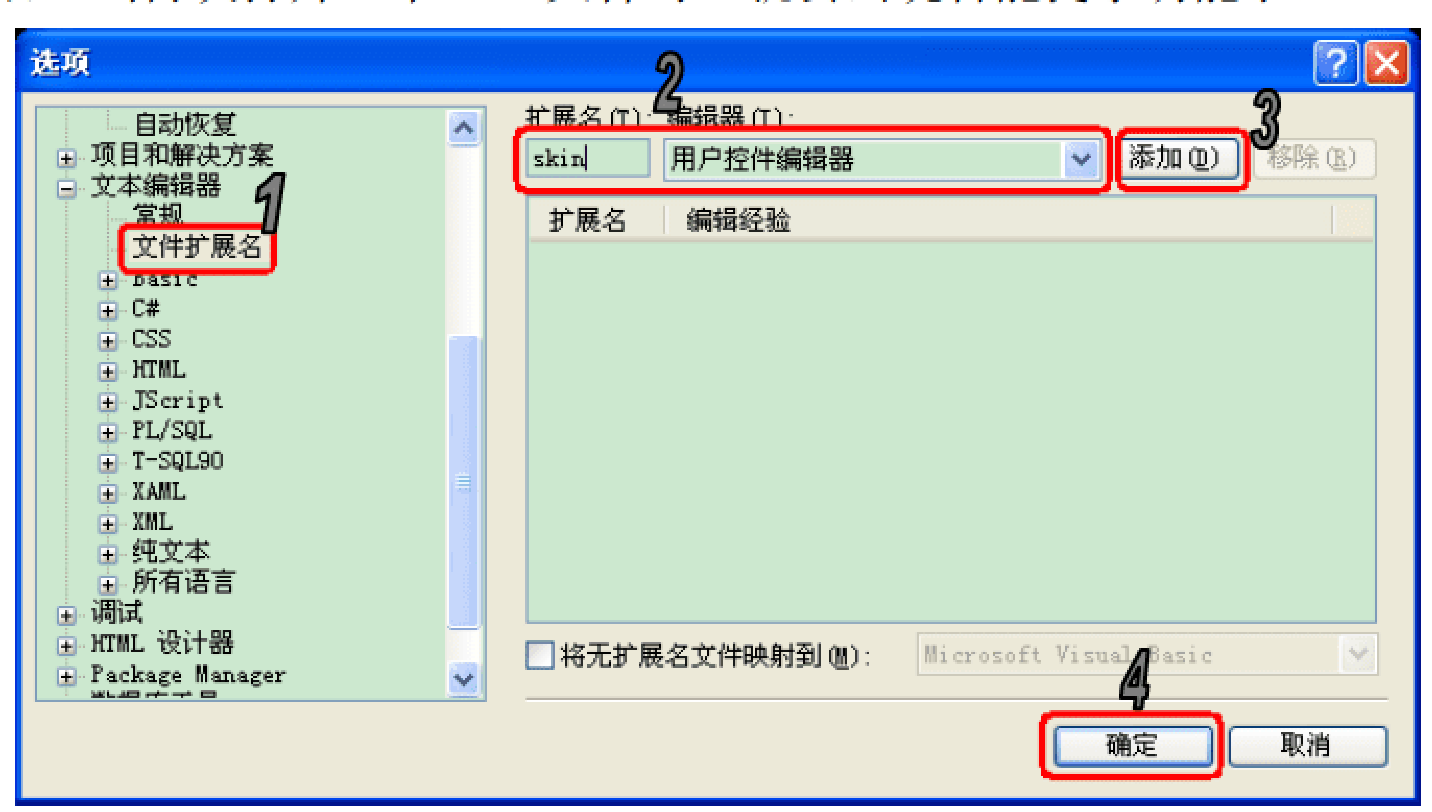


图 4-18 【选项】对话框

知识点

也可以不在外观文件中直接编写定义控件外观的代码，而是先在页面中设置控件的属性，然后再将自动生成的代码复制到外观文件中，去掉所有控件的 ID 属性即可。

【例 4-3】创建一个包含一些简单外观的主题，这些外观用于定义控件的外观。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 4-3】。



(2) 在【解决方案资源管理器】中，右击项目名，从弹出的快捷菜单中选择【添加】|【添加 ASP.NET 文件夹】|【主题】命令，系统将创建名为 App_Themes 的文件夹和名为“主题 1”的子文件夹。

(3) 右击“主题 1”文件夹，从弹出的快捷菜单中选择【添加新项】命令，添加一个新的外观文件 SkinFile.skin。

(4) 新添加的文件中包含一段外观文件编写的说明文字和两个示例，如下所示：

```
<%--
默认的外观模板。以下外观仅作为示例提供。
1. 命名的控件外观。SkinId 的定义应唯一，因为在同一主题中不允许一个控件类型有重复的 SkinId。
<asp:GridView runat="server" SkinId="gridviewSkin" BackColor="White" >
    <AlternatingRowStyle BackColor="Blue" />
</asp:GridView>
2. 默认外观。未定义 SkinId。在同一主题中每个控件类型只允许有一个默认的控件外观。
<asp:Image runat="server" ImageUrl="~/images/image1.jpg" />
--%>
(5) 按照以上示例，添加如下代码，定义 Label 和 Button 控件的外观。
<asp:Label runat="server" ForeColor="red" Font-Size="14pt" Font-Names="Verdana" />
<asp:Button runat="server" Borderstyle="Solid" Borderwidth="2px" Bordercolor="Blue" Backcolor="yellow"/>
```



(6) 保存该外观文件。打开 Default.aspx 文件，切换到设计视图，添加一个 Label 控件和一个 Button 控件。

(7) 在【属性】面板中选择 Document 元素，设置 Theme 属性为“主题 1”，切换到源视图中，可发现代码第一行的@ Page 指令中添加了如下属性：

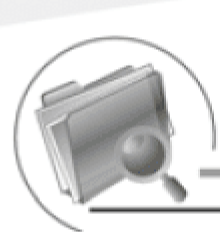
```
<%@ Page … Theme="Theme1"%>
```

(8) 编译并运行程序，在默认浏览器中打开该页面，查看设置效果，如图 4-19 所示。

如果希望某些控件的外观和页面中具有相同类型的其他控件的外观不一样，则可以在.skin 文件中给特定的控件添加一个 SkinID 属性，例如，在【例 4-3】中增加一个按钮，其外观定义如下：

```
<asp:Button runat="server" SkinID="GreenBtn"   Font-Size="14pt" Borderstyle="dotted" Borderwidth="2px"
Bordercolor="red" Backcolor="Green"/>
```

在 Default.aspx 页面中再添加一个 Button 控件，设置其 SkinID 属性为 GreenBtn。显示效果如图 4-20 所示。



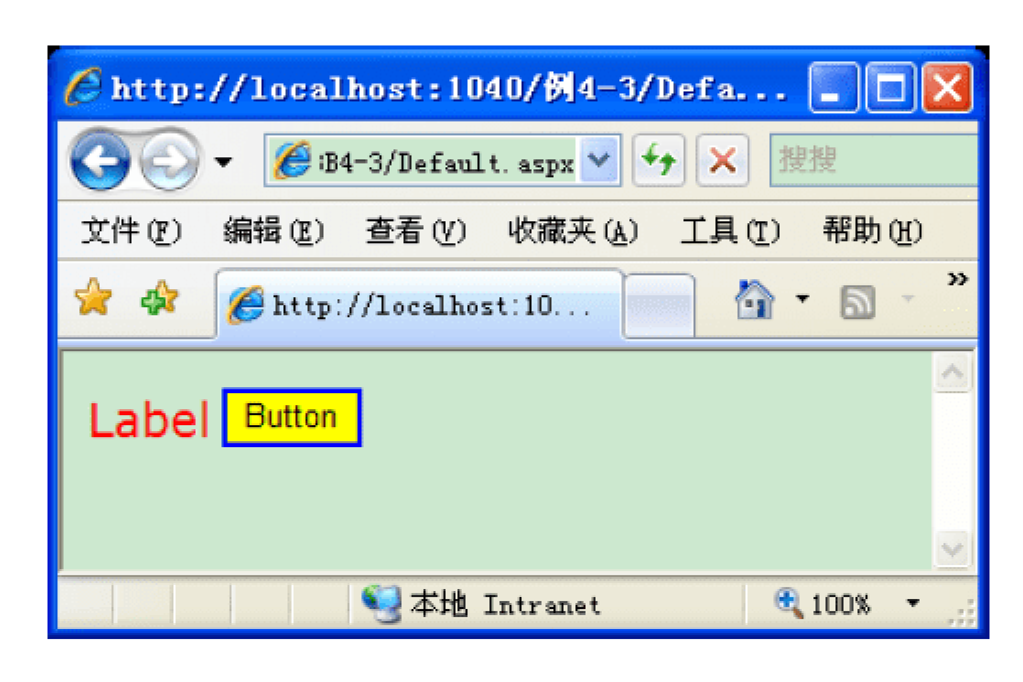


图 4-19 应用外观后的控件

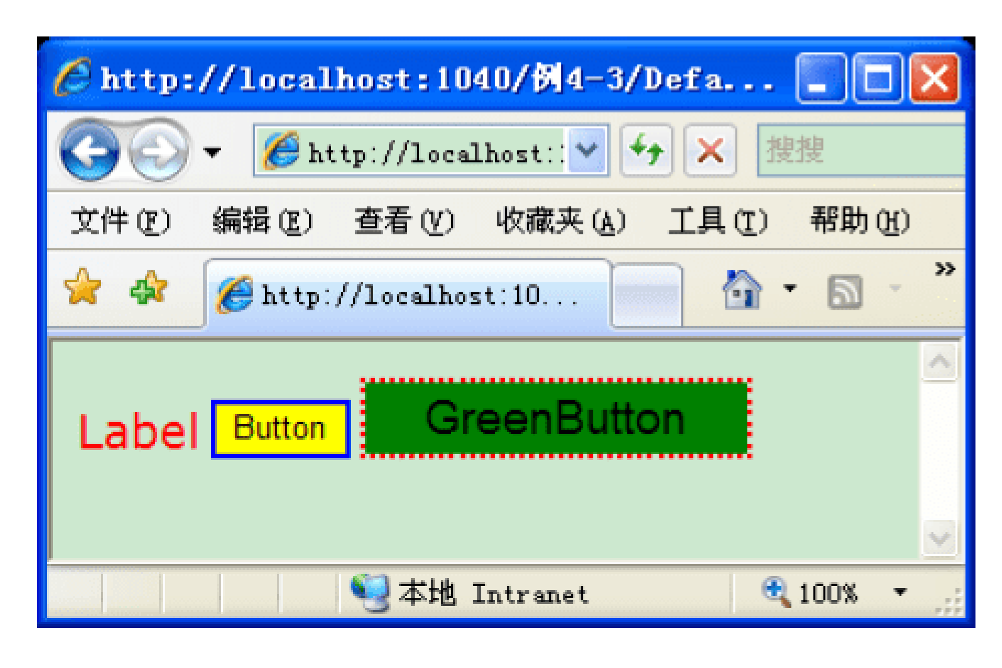


图 4-20 应用 SkinID 属性的控件

2. 在主题中定义样式表

除了外观文件，在主题中还可以定义.css 文件，然后在网页文件中设置 StyleSheetTheme 属性为定义的主题即可。

【例 4-4】在网页文件中同时使用外观文件和样式表文件。

(1) 启动 VWD 2010，打开网站【例 4-3】。

(2) 右击【主题 1】文件夹，从弹出的快捷菜单中选择【添加新项】命令，添加一个样式表文件 StyleSheet.css。

(3) 在 StyleSheet.css 样式文件中添加如下代码，定义 h1 的样式：

```
h1
{
    border-style: dashed;
    font-size: 1.6em;
    padding-bottom: 0px;
    margin-bottom: 0px;
    color: #FF0000;
}
```

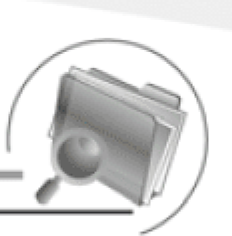
(4) 在 Default.aspx 页面中添加<h1>标记的元素：

```
<h1>欢迎光临小石头网站</h1>
```

(5) 修改当前页面的 Document 标记中属性 StyleSheetTheme 的值为“主题 1”。

(6) 编译并运行程序，在浏览器中即可看到引入外观和样式表文件后的最终显示效果，如图 4-21 所示。





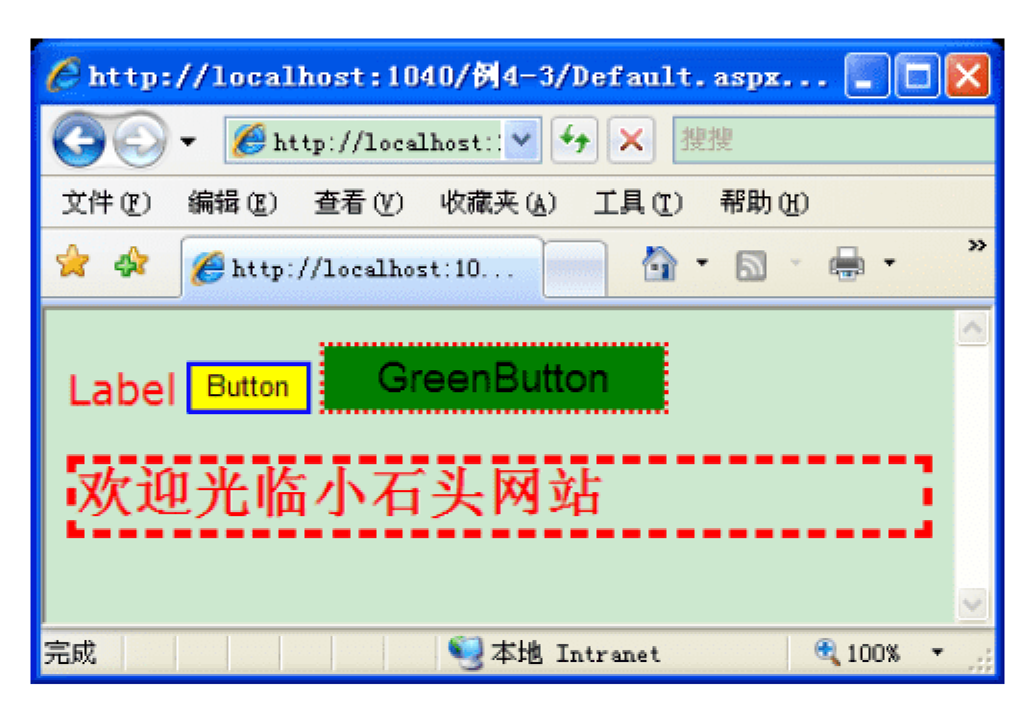


图 4-21　引入外观和样式后的页面效果

创建了主题之后，可以定制如何在应用程序中使用主题，方法是：将主题作为自定义主题与网页文件关联，或者将主题作为样式表主题与网页文件关联。样式表主题和自定义主题都使用相同的主题文件，但是样式表主题在网页文件的控件和属性中的优先级最低。在 ASP.NET 中，优先级的顺序是：

(1) 主题设置，包括 Web.config 文件中设置的主题。

(2) 本地网页文件的样式属性设置。

(3) 样式表主题设置。

在这里，如果选择使用样式表主题，则在网页文件中本地声明的任何样式信息都将覆盖样式表主题的属性。同样，如果使用自定义主题，则主题的属性将覆盖本地网页文件中设置的任何样式内容，以及使用中的任何样式表主题中的任何内容。



3. 主题的应用级别

有 3 个不同的选项可以向 Web 站点应用主题，分别是：在 Page 指令中的页面级、在站点级修改 Web.config 文件，以及通过程序来设置主题。

⦿　在页面级设置主题

在例 4-3 和例 4-4 中就是使用这种方式来应用主题的。在页面级设置 Theme 属性或 StyleSheetTheme 属性很容易，只要设置页面的 Page 指令中的相关属性即可。

```
<%@ Page Language="C#" AutoEventWireup="false" CodeFile="Default.aspx.cs"
        Inherits="_Default" Theme="主题 1" StyleSheetTheme="主题 1" %>
```

用 StyleSheetTheme 替换 Theme 来应用一个主题，该主题的设置可以由单个页面重写。

⦿　在站点级设置主题

为了在整个 Web 站点中强制应用同一个主题，可以在 Web.config 文件中设置主题。要做到这一点，需要将一个 theme 属性添加到<system.web>元素内的<pages>元素中：

```
<pages theme="主题 1">
    ...
</pages>
```

确保全部用小写字母输入 theme，因为 Web.config 文件中的 XML 是区分大小写的。

◉ 通过程序来设置主题

设置主题的第三种也是最后一种方式是通过代码来编程设置。由于主题的工作方式，需要在页面生命周期的早期完成这一工作。通常是在 PreInit 事件通过 Page 对象的 Theme 属性来设置主题。

4.3.4 动态切换主题

动态切换主题是指在运行时切换主题。例如，可以通过允许用户使用喜欢的颜色和布局。由于使用的是在运行时向页面应用主题的方式，因此需要在页面的生命周期较早的时候设置主题，即在 PreInit 事件中设置。

为了允许用户修改主题，可以向其提供一个下拉菜单，当用户修改列表中的活动选项时，该菜单自动向服务器发起回发请求。在服务器上，就会得到从列表中选择的主题，将它应用到页面上，然后将选项存储在 Cookie 中，以便在下次访问 Web 站点时检索它。

【例 4-5】让用户选择自己喜欢的主题，实现动态换肤功能。

(1) 启动 VWD 2010，新建网站【例 4-5】。

(2) 在【解决方案资源管理器】中，右击项目名，从弹出的快捷菜单中选择【添加】|【添加 ASP.NET 文件夹】|【主题】命令，系统将创建名为 App_Themes 的文件夹和名为“主题 1”的子文件夹。

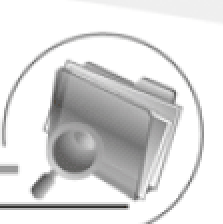
(3) 右击“主题 1”文件夹，从弹出的快捷菜单中选择【添加新项】命令，添加一个新的外观文件 SkinFile.skin，用同样的方法再在该文件夹下添加一个 CSS 样式文件 StyleSheet.css。

(4) 右击“主题 1”文件夹，从弹出的快捷菜单中选择【新建文件夹】命令，新建一个文件夹，修改其名称为 images，在 images 文件夹上右击，选择【添加现有项】命令，添加一个图片作为主题 1 的背景图片。

(5) 用相同的操作创建“主题 2”，其中也包括一个外观文件 SkinFile.skin、一个 CSS 样式文件 StyleSheet.css 以及 images 文件夹和“主题 2”的背景图片，目录结构如图 4-22 所示。

(6) “主题 1”在这里定义为“天蓝蓝”，所以背景图片和控件的外观都以蓝色为主，在“主题 1”的外观文件中添加如下代码:

```
    <asp:Label runat="server" BackColor="#CCCCFF" Font-Bold="True" Font-Size="Large" ForeColor="#CC00FF"
BorderStyle="Dashed" />
      <asp:DropDownList runat="server"   BackColor="White" Font-Bold="True"
                  ForeColor="#CC33FF" Font-Size="Large"/>
      <asp:Calendar runat="server" BackColor="White"
                  BorderColor="#3366CC" BorderWidth="1px" CellPadding="1"
                  DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
```



```
            ForeColor="#003399" Height="200px" Width="220px">
               <DayHeaderStyle BackColor="#99CCCC" ForeColor="#336666" Height="1px" />
               <NextPrevStyle Font-Size="8pt" ForeColor="#CCCCFF" />
               <OtherMonthDayStyle ForeColor="#999999" />
               <SelectedDayStyle BackColor="#009999" Font-Bold="True" ForeColor="#CCFF99" />
               <SelectorStyle BackColor="#99CCCC" ForeColor="#336666" />
               <TitleStyle BackColor="#003399" BorderColor="#3366CC" BorderWidth="1px"
                    Font-Bold="True" Font-Size="10pt" ForeColor="#CCCCFF" Height="25px" />
               <TodayDayStyle BackColor="#99CCCC" ForeColor="White" />
               <WeekendDayStyle BackColor="#CCCCFF" />
          </asp:Calendar>
```

(7) “主题 1”的样式表文件中只定义背景图片，所以添加如下代码即可：

```
body
{
     background-image: url('images/blue.jpg');
}
```



(8) “主题 2”定义为“夕阳红”，其背景图片和控件的外观都以红色为主，在“主题 1”的外观文件中添加如下代码：

```
     <asp:Label runat="server" BackColor="White" Font-Bold="True" Font-Size="Large" ForeColor="Red"
BorderStyle="Groove" />
     <asp:DropDownList   runat="server"   BackColor="White" Font-Bold="True"
               ForeColor="Red" Font-Size="Large"/>
     <asp:Calendar runat="server" BackColor="#FFFFCC"
               BorderColor="#FFCC66" BorderWidth="1px" DayNameFormat="Shortest"
               Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399" Height="200px"
               ShowGridLines="True" Width="220px">
               <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
               <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
               <OtherMonthDayStyle ForeColor="#CC9966" />
               <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
               <SelectorStyle BackColor="#FFCC66" />
               <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt"
                    ForeColor="#FFFFCC" />
               <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
          </asp:Calendar>
```

(9) 在“主题 2”的样式表文件中添加如下代码:

```
body
{
    background-image: url('images/sunset.jpg');
}
```

(10) 打开 Default.aspx 页面，切换到设计视图，添加一个 Label 控件、一个 DropDownList 控件和一个 Calendar 控件，只需设置 Label 控件的 Text 属性，DropDownList 控件的 Item、AutoPostBack 属性即可，生成的代码如下:

```
<asp:Label ID="Label1" runat="server" Text="请选择主题："></asp:Label>
<asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True">
    <asp:ListItem Value="主题 1">天蓝蓝</asp:ListItem>
    <asp:ListItem Value="主题 2">夕阳红</asp:ListItem>
</asp:DropDownList>
```


(11) 为 DropDownList 控件添加 SelectedIndexChanged 事件处理程序，代码如下:

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    HttpCookie myTheme = new HttpCookie("myTheme");
    myTheme.Expires = DateTime.Now.AddMonths(3);
    myTheme.Value = DropDownList1.SelectedValue;
    Response.Cookies.Add(myTheme);
    Response.Redirect(Request.Url.ToString());
 }
```

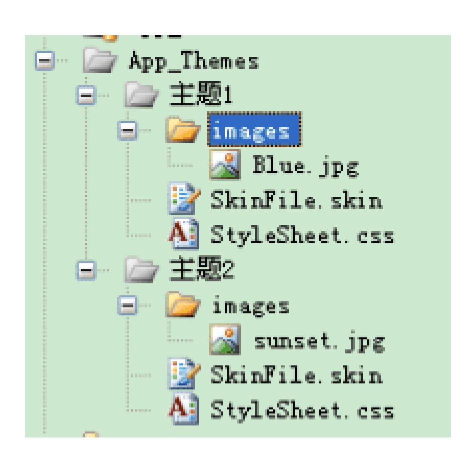


图 4-22　主题的目录结构

知识点

在下拉列表的事件处理程序中，最后一步是将用户重定向到同一个页面，否则就不会立即应用新主题。因为主题需要在页面的生命周期的早期设置，所以不能再为当前的请求设置。通过将用户重定向到同一个页面，就发出了一个能够成功应用选中主题的新请求。

(12) 当页面加载时将需要再次从列表中预先选择恰当的项，以显示正确的主题。进行此操作的最佳位置是在 Page 类的 Load 事件中。添加处理程序的代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
```



```
        {
            string selectedTheme = Page.Theme;
            HttpCookie myTheme = Request.Cookies.Get("myTheme");
            if (myTheme != null)
            {
                selectedTheme = myTheme.Value;
            }
            if (!string.IsNullOrEmpty(selectedTheme) &&
                       DropDownList1.Items.FindByValue(selectedTheme) != null)
            {
                DropDownList1.Items.FindByValue(selectedTheme).Selected = true;
            }
        }
    }
```

(13) 正如前面所提到的，主题需要在 PreInit 事件(该事件在页面生命周期的早期发生)中设置。在该事件内可以查看带选中主题的 cookie 是否存在。如果存在，就可以用它的值设置恰当的主题，代码如下：

```
protected void Page_PreInit(object sender, EventArgs e)
{
    HttpCookie preferredTheme = Request.Cookies.Get("myTheme");
    if (preferredTheme != null)
    {
        Page.Theme = preferredTheme.Value;
    }
}
```

(14) 编译并运行程序，在浏览器中打开 Default.aspx 页面，通过下拉列表选择不同的主题，效果如图 4-23 所示。



图 4-23　动态切换主题效果



4.4 母版页

在构建 Web 站点时，应该努力使布局和行为尽可能保持一致。而且有很多元素，如站点标题、公共导航以及版权信息等，会出现在每一个页面中，这些元素的一致布局会让用户知道自己始终是在同一个站点中。虽然这些元素可以通过在 XHTML 中使用包含文件构建，但 ASP.NET 4.0 和 VWD 2010 提供了更加健壮的母版页技术来实现。

母版页的最大好处是它们可以在单个地方定义站点中所有页面的全局外观。这意味着如果要修改站点的布局——比如要把菜单从左边移到右边——只需修改母版页，基于此类母版页的页面就会自动进行相应的修改。

母版页是用于设置页面外观的模板，是一种特殊的 ASP.NET 网页文件，同样也具有其他 ASP.NET 文件的功能，如添加控件、设置样式等，只不过扩展名是.master。在母版页中，界面被分为公用区和可编辑区，公用区的设计方法与一般页面的设计方式相同，可编辑区用 ContentPlaceHolder 控件预留出来。

引用母版页的.aspx 页面称为内容页，在内容页中，母版页的 ContentPlaceHolder 控件预留的可编辑区会被自动替换为 Content 控件，开发人员只需要在 Content 控件区域中填充内容即可，在母版页中定义的其他标记将自动出现在引用该母版页的.aspx 页面中，母版页的部分以灰色显示，表示不能修改这些内容。

每一个母版页中可以包含一个或多个内容页。使用母版页可以统一管理和定义具有相同布局风格的页面，给网页设计和修改带来极大的方便。使用母版页有如下优点：

- 使用母版页可以集中处理页的通用功能，以便可以只在一个位置进行更新。
- 使用母版页可以方便地创建一组控件和代码，并将结果应用于一组新的页面。
- 通过允许控制占位符控件的呈现方式，母版页可以在细节上控制最终页的布局。
- 母版页提供一个对象模型，使用该对象模型可以从各个内容页自定义母版页。

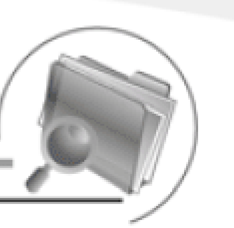


提示

在使用母版页时，母版页中使用的图片和超链接应尽量使用服务器端控件来实现，如 Image 和 HyperLink 控件。即使控件不需要服务器代码也是如此，这是因为将设计好的母版页或内容页移动到另一个文件夹时，如果使用的是服务器控件，即使不改变服务器控件的 URL，ASP.NET 也可以正确解析，并自动将其 URL 改为正确的位置，但是如果使用了普通 HTML 标记，那么 ASP.NET 将无法正确解析这些标记的 URL，从而导致图片不能显示和链接失败，给维护带来极大麻烦。

4.4.1 创建母版页

当创建新的 Web 站点时，总是先添加作为所有其他页面的基础的母版页，即使站点中只有



少数几个页面，母版页仍然可以帮助确保整个站点拥有一致的外观。

在某种程度上，母版页看起来就像正常的 ASPX 页面。创建母版页的方法也和创建一般页面的方法非常相似，区别是母版页无法单独在浏览器中查看，必须通过创建内容页才能浏览。

1. 创建母版页

下面这个例子是一个很常见的布局，母版页中包含一个标题、一个导航菜单和一个页脚，这些内容将在站点的每个页面中出现。在母版页中包含两个内容占位符，其中导航菜单有默认内容，主区域为空，这是母版页中的一个可变区域，可以使用内容页中的信息来替换此区域。

【例 4-6】创建一个母版页。

(1) 启动 VWD 2010，新建网站【例 4-6】。

(2) 在【解决方案资源管理器】中，右击网站的名称，从弹出的快捷菜单中选择【添加新项】命令，在打开的【添加新项】对话框中选择【母版页】模板，添加名为 MasterPage.master 的母版页。



(3) 观察母版页的源代码，在页面的顶部是一个@ Master 声明，而不是通常在 ASP.NET 页面中看到的@ Page 指令，它也有 CodeFile 和 Inherits 属性，如下所示：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
```

(4) 此外，页面的主体还包含一个 ContentPlaceHolder 控件，这是母版页中的一个区域，其中的可替换内容将在运行时由内容页合并。为了方便母版页的编辑，通常情况下先将 ContentPlaceHolder 控件删除，母版页编辑完成后再放置 ContentPlaceHolder 控件，下面的步骤将采用这种方法布局。



提示

尽管通常只需要几个占位符就可以创建灵活的页面布局，但实际上想创建多少就可以创建多少。

(5) 在母版页的<form>标记之间添加下面的代码，替换<div>标记与创建母版页时 VWD 添加的 ContentPlaceHolder。

```
<form id="form1" runat="server">
  <div id="PageWrapper">
    <div id="top" align="center"
          style="background-color:#FF6600; font-family:  微软雅黑; color: #FFFFFF;"><h1>欢迎光临小石
头网站</h1></div>
    <div id="menu" align="right">
      <asp:ContentPlaceHolder id="menuContent" runat="server">
        <a href="Default.aspx">首页</a> <a href="link.aspx">友情链接</a> <a href="About.aspx">关
于本网站</a>
```

```
            </asp:ContentPlaceHolder>
        </div>
        <div id="main">
            <asp:ContentPlaceHolder ID="mainContent" runat="server">
            </asp:ContentPlaceHolder>
        </div>
        <div id="footer" align="center" style="color:Gray">版权所有(C)小石头网站 2011.07.30</div>
    </div>
</form>
```

(6) 现在已经创建了母版页，所以可以先保存并关闭母版页。该母版页的设计视图效果如图 4-24 所示。

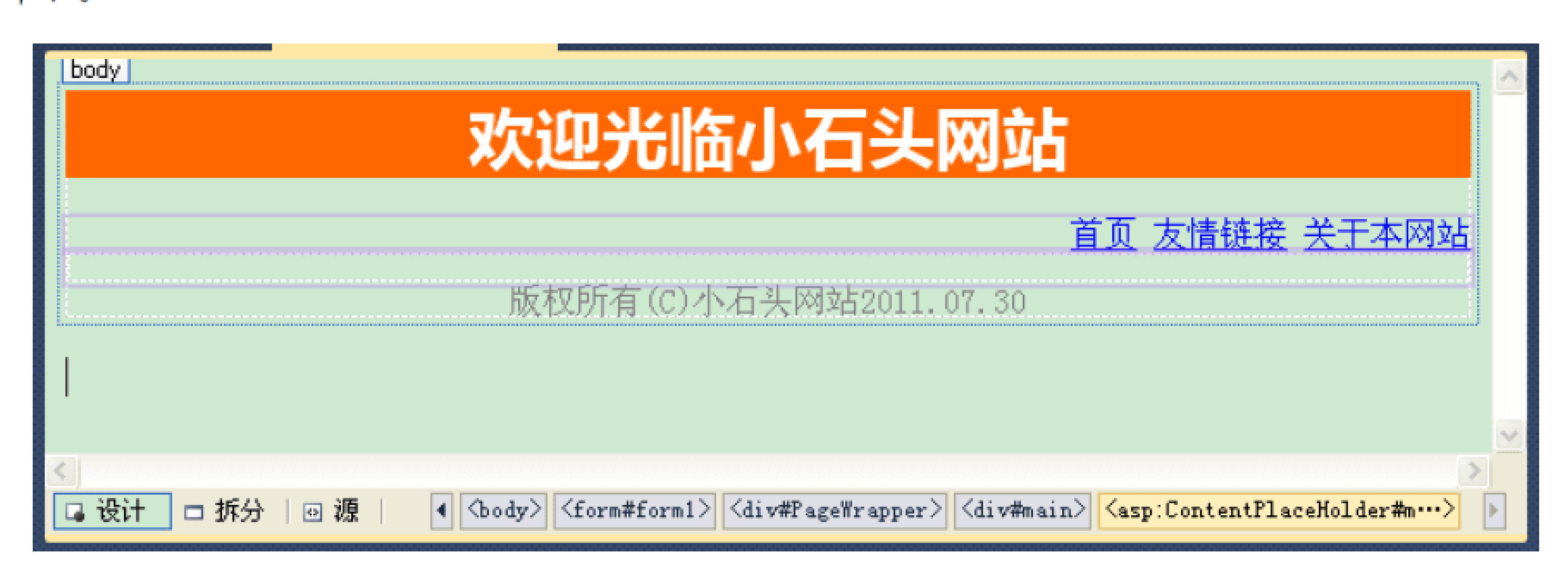


图 4-24　母版页设计效果

在下一小节，将会看到如何将该母版页面作为内容页面的模板使用。

2. 母版页详解

前面已经创建了带有主内容占位符的母版页。切换到母版页的源视图，可发现页面的页头 head 部分也有一个 Content PlaceHolder。

```
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
```

每当创建一个新的母版页时都会自动添加此占位符，在内容页中可以用它来添加页面特有的位于页面的<head>标记之间的内容，比如 CSS(包括内嵌样式表和外部样式表)和 JavaScript。

母版页中名为 menuContent 的 ContentPlaceHolder 包含 3 个超链接，这是可以作为内容页的默认新项，当基于该母版页新建页面时，内容页即可以重写这部分内容，也可以不重写。



4.4.2　创建内容页

母版页如果没有内容页来使用它，就没有用处。通常仅有少量几个母版页，却可以有很多内容页。为了将一个内容页基于一个母版页，可以在添加新网页到站点时，选中【添加新项】对话框底部的【选择母版页】复选框；也可以在直接在页面上设置 MasterPageFile 属性。

内容页中只能含有映射到母版页中的<asp:ContentPlaceHolder>控件的<asp:Content>控件。而这些控件又可以包含标准标记，比如 HTML 和服务器控件声明。因为内容页中的整个标记需要用<asp:Content>标记括起来，所以不太容易将现有 ASPX 页面转换为内容页。通常是将要保留的内容复制到剪贴板上，删除原页面，然后基于母版页添加新页面。添加了该页面后，再把剪贴板上的内容粘贴到<asp:Content>标记内。

【例 4-7】基于【例 4-6】创建的母版页创建内容页。

(1) 启动 VWD 2010，打开网站【例 4-6】。

(2) 在【解决方案资源管理器】中，删除 Default.aspx 页面。然后添加 3 个新页面: Defualt.aspx、link.aspx、About.aspx。添加上述 3 个页面时，需要选中【添加新项】对话框中的【选择母版页】复选框，并在弹出的【选择母版页】对话框中选择之前创建的母版页 MasterPage.master。

(3) 基于母版页新建的网页初始代码如下所示:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
CodeFile="Default.aspx.cs" Inherits="_Default" %>
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="menuContent" Runat="Server">
</asp:Content>
<asp:Content ID="Content3" ContentPlaceHolderID="mainContent" Runat="Server">
</asp:Content>
```

(4) 指向 ContentPlaceHolder 的 Content 控件的 ContentPlaceHolderID 属性是在母版页中定义的。ContentPlaceHolderID 为 head 的占位符就是用来添加页面特有的位于<head>标记之间的内容的，本例中对此占位符不做任何修改，只设置菜单内容和主内容区域。

(5) 切换到 Default.aspx 页面的设计视图，单击 menuContent 控件右侧的小三角按钮，打开【Content 任务】面板，选择【默认为母版页的内容】选项，此时将弹出【确认】对话框，提示用户如果使用母版页的内容将从网页中删除此区域中的所有内容，单击【是】按钮，如图 4-25 所示。

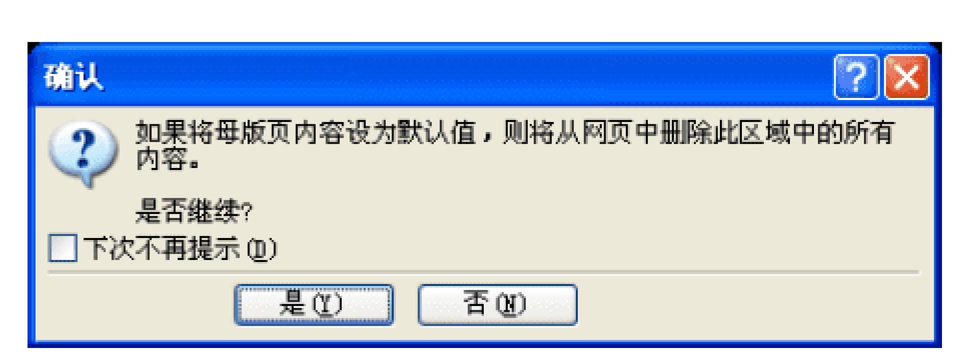


图 4-25　【确认】对话框




知识点

将默认值设置为母版页的内容之后，还可以通过【Content 任务】面板中的【创建自定义内容】选项来再次创建自己的内容。

(6) 分别在 3 个页面的 mainContent 区域添加不同的内容以区分不同的页面。

(7) 编译并运行程序，当在浏览器中请求基于母版页的页面时，服务器会阅读内容页与母版页，将两者合并，然后将最终结果发送给浏览器，效果如图 4-26 所示。

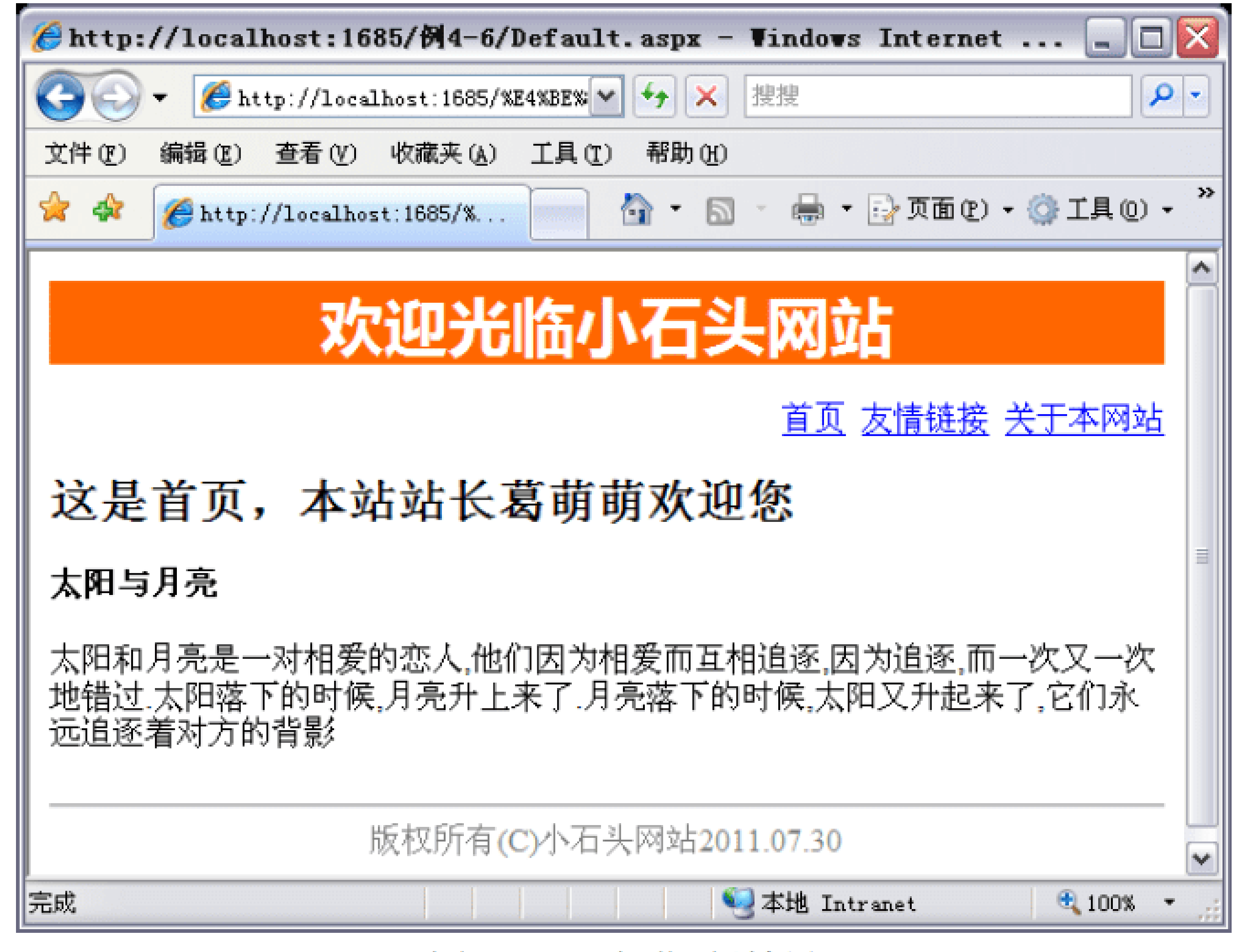


图 4-26 页面运行效果


提示

母版页也可以嵌套。嵌套母版页是基于另一个母版页的母版页。内容页面则可以基于嵌套母版页。如果有一个目标为不同区域仍然需要共享相同外观的 Web 站点，采用嵌套母版页就比较有用。

4.5 上机练习

本章的上机实验主要练习在内容页中访问母版页的成员。从而实现母版页与内容页的信息交换。在内容页中可以通过编程方式访问母版页中的成员，包括母版页上的任何公共属性或方法以及任何控件。要实现内容页对母版页中定义的属性或方法进行访问，则该属性或方法必须声明为公共成员(public)。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 4】。

(2) 添加母版页 myMaster.master，在【解决方案资源管理器】中右击 myMastermaster，从弹出的快捷菜单中选择【查看代码】命令，打开其后台代码文件 myMaster.master.cs。

(3) 在类定义中创建名为 strName 的属性，并在视图状态中存储该属性的值。添加的代码如下：



```
public string strName
{
    get{return (string)ViewState["myName"];}
    set { ViewState["myName"] = value; }
}
```

(4) 添加页面的 Init 事件处理程序代码，如下：

```
void Page_Init(Object sender, EventArgs e)
{
    this.strName = "葛萌萌";
}
```

(5) 基于该母版页创建内容页 myPage.aspx，并切换到该页面的源视图。在页面顶部的@ Page 指令下面添加@ MasterType 指令：

```
<%@ MasterType virtualpath="~/myMaster.master" %>
```

该指令的作用是将内容页的 Master 属性绑定到 myMaster.master 页。

(6) 切换到该页的设计视图，在 Content 控件中添加一个 Label 控件。

(7) 在 myPage.aspx 页面的 Load 事件处理程序中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "这是获取到母版页中的变量值：" + Master.strName;
}
```

(8) 测试内容页，切换 myPage.aspx 页，然后按【Ctrl+F5】组合键运行页面。在默认浏览器中打开该页面，显示效果如图 4-27 所示。

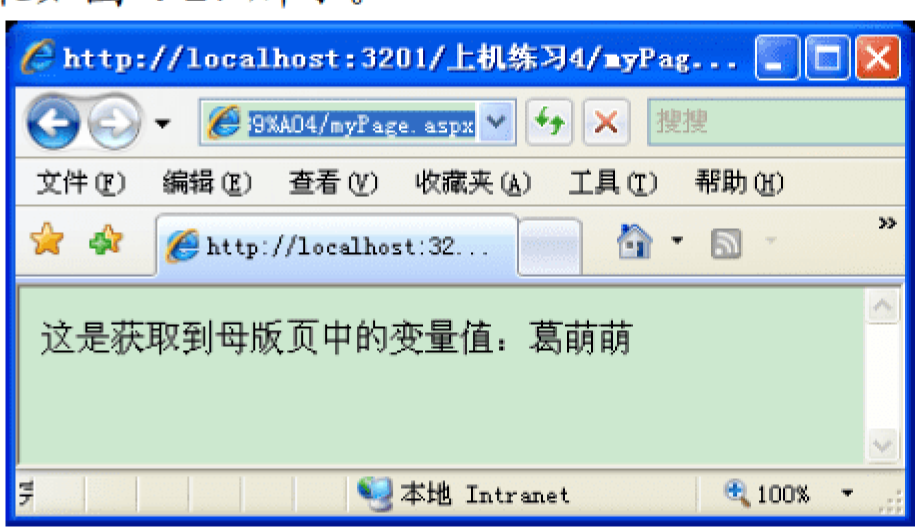


图 4-27　获取母版页中的变量



4.6　习题

1. VWD 提供了哪些使用 CSS 的便利工具？
2. 在下面两个规则中，哪个规则比较容易在 Web 站点中跨页面重用？请解释原因。

```
#MainContent
{
  border: 1px solid blue;
}
.BoxWithBorders
{
  border: 1px solid blue;
}
```

3. 解释设置页面的 Theme 属性与 StyleSheetTheme 属性之间的区别。
4. 当控件的属性和主题中控件的外观定义发生冲突时，哪个有较高优先级？
5. 如何将内容页中的 Content 控件与母版页中的 ContentPlaceHolder 关联起来？
6. 如何禁用主题？
7. 创建一个 CSS 规则，将站点中所有的一级标题(h1)的外观设置为：

- 字体使用 Arial，并且加粗；
- 颜色为红色；
- 字体大小为 18 像素；
- 上边框和左边框为蓝色细边。



第5章 显示和操作数据库

学习目标

ASP.NET 应用程序的数据访问是通过 ADO.NET 进行的，ADO.NET 可以使 Web 应用程序从各种数据源中快速访问数据。本章首先介绍数据库的基本知识和SQL语言，接着介绍ADO.NET访问数据库的方法，最后介绍了 ASP.NET 提供的数据绑定技术和数据控件的使用，通过本章的学习读者应该掌握如何访问和操作数据源，以及数据信息的显示与更新。

本章重点

- 掌握使用 SQL 来操作数据
- 了解 ADO.NET 的基本知识
- 掌握 ADO.NET 访问数据库的方法
- 掌握单值和列表控件的数据绑定
- 理解数据源控件的工作原理
- 掌握 GridView 控件的使用方法和技巧
- 学会设计主-从页面显示数据库信息

5.1 数据库基础

数据库是非常有用的，因为它允许通过结构化的方式来存储和检索数据。数据库最大的好处是能够在运行时被访问，这就意味着在 VWD 中，将不再局限于在设计时所创建的相对静态文件。

5.1.1 数据库概述

数据库就是数据的集合，例如，日常生活中，我们用笔记本记录亲戚和朋友的联系方式，将

其姓名、地址、电话等信息都记录下来。这个“通讯录”就是一个最简单的“数据库”，每个人的姓名、地址、电话等信息就是这个数据库中的“数据”。

最为流行的一种数据库是关系数据库(Relational Database)。这种数据库常用于 Web 站点中，也将用于本书后续部分。不过，关系数据库并不是唯一的数据库类型，还有其他类型的数据库，包括平面文件数据库、对象关系数据库和面向对象数据库，但这些数据库在 Internet 应用程序中不常见。

关系数据库中有表(table)的概念，其中数据以行和列的形式存储，如同电子表格一样。表中的每行包含存储于其中的记录项的完整信息，而每列包含表中记录项的特定属性的信息。

“关系”指的是数据库中不同表相互关联的方式。它不是将相同的数据一遍遍地复制，而是在其自己的表中存储重复的数据，然后从其他表中创建与该数据的关系。

在 ASP.NET 项目中可以使用多种不同类型的数据库，包括 Microsoft Access、SQL Server、Oracle 和 MySQL。不过，在 ASP.NET 4.0 Web 站点中最常用的数据库是 Microsoft 的 SQL Server。本书主要使用 Microsoft SQL Server 2008 Express Edition，因为它是随 VWD 免费提供的，有着许多创新性的功能。而且，由于其数据库引擎与 SQL Server 2008 商业版的相同，因而可以在开发周期的后续阶段轻松地升级到那些版本。

本书所用的关系数据库是 Microsoft SQL Server 2008 R2，使用的数据库是下面新建的信息管理数据库 InfoManage，该数据库中包含 4 个用户表：Student(学生表)、Class(班级表)、Score(成绩表)、Course(课程表)。各表的字段信息如表 5-1~表 5-4 所示，表之间的关系如图 5-1 所示。

表 5-1　Student(学生表)字段信息

字　段　名	字 段 描 述	数 据 类 型	备　　注
Sno	学号	int	主键
Sname	学生姓名	nvarchar(20)	非空
Sgender	性别	char(1)	非空
Stelephone	联系电话	nvarchar(20)	可空
Saddress	地址	nvarchar(50)	可空
Cno	班级号	int	外键

表 5-2　Class(班级表)字段信息

字　段　名	字 段 描 述	数 据 类 型	备　　注
Cno	班级号	int	主键
Cname	班级名称	nvarchar(50)	非空

表 5-3　Score(成绩表)字段信息

字　段　名	字 段 描 述	数 据 类 型	备　　注
Sno	学号	int	外键
degree	成绩	int	非空
courseID	课程 ID	int	外键

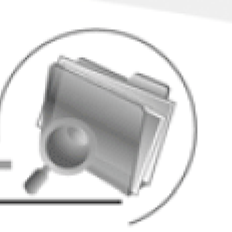


表 5-4　Course(课程表)字段信息

字　段　名	字 段 描 述	数 据 类 型	备　　注
courseID	课程 ID	int	主键
courseName	课程名称	nvarchar(50)	非空

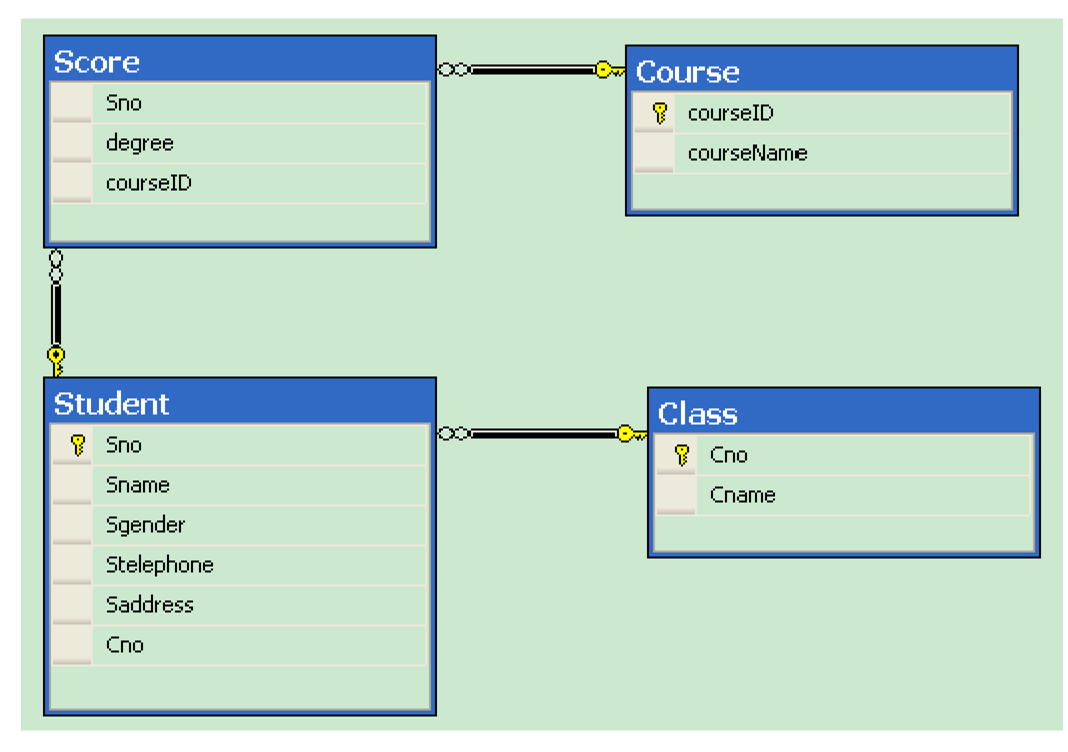


图 5-1　表之间的关系



5.1.2　结构化查询语言 SQL

为了成功地在 ASPX 页面中运用数据库，下面将来学习如何使用 SQL(Structured Query Language，结构化查询语言)查询语言来访问数据库，该语言允许检索和操纵存储在数据库中的数据。

1. SQL 概述

SQL 语言是一种介于关系代数和关系演算之间的结构化查询语言，其功能并不仅仅是查询，还具备数据定义和数据操纵等功能。ANSI(美国国家标准协会)规定 SQL 为关系型数据库管理系统的标准语言。目前，绝大多数流行的关系型数据库管理系统，如 Oracle、Sybase、Microsoft SQL Server、Access 等，都采用了 SQL 语言标准。

Microsoft SQL Server 2008 数据库支持 ANSI 92 SQL 标准中定义的大部分语法。在这一标准之上，Microsoft 又添加了一些专用扩充，合起来称为 T-SQL(Transact SQL)。本书后续部分将使用 SQL 这一术语。

总的来说，SQL 语言具有以下特点：

- 类似于英语自然语言，简单易学。
- 是一种非过程语言。
- 是一种面向集合的语言。
- 既可独立使用，又可嵌入到宿主语言中使用。
- 具有查询、操纵、定义和控制一体化功能。

SQL 语言包含以下 4 个部分：

- 数据定义语言(DDL，Data Definition Language)：CREATE、ALTER、DROP。
- 数据查询语言(DQL，Data Query Language)：SELECT。
- 数据操纵语言(DML，Data Manipulation Language)：INSERT、UPDATE、DELETE。
- 数据控制语言(DCL，Data Control Language)：COMMIT、ROLLBACK。

对于数据库编程人员来说，与数据库交互时，使用最多的就是查询和操纵数据操作，所以本书重点介绍 SELECT、INSERT、UPDATE、DELETE 这 4 个语句。


提示

SQL 语言不区分大小写，SELECT 与 Select 的含义是相同的。为了统一起见，本书中所书写的 SQL 命令全部使用大写形式。



2. SELECT 语句

要从数据库中读取数据，首先需要表明要从查询的表中检索哪些列，这是通过 SELECT 语句完成的；接着需要使用 FROM 关键字表明想从什么表中获得数据；然后需要筛选数据，确保只返回符合条件的记录，可以使用 SQL 语句中的 WHERE 子句筛选数据；最后，可以使用 ORDER BY 子句对结果进行排序。

完整的 SELECT 语句格式如下所示，其中方括号中的子句都是可选的：

```
SELECT 目标表的列名或列表达式集合
FROM 基本表或(和)视图集合
［WHERE 条件表达式］
［GROUP BY 列名集合
［HAVING 组条件表达式］］
［ORDER BY 列名［集合］…］
```

首先从 FROM 子句列出的表中，选择满足 WHERE 子句给出的条件表达式的记录，然后按 GROUP BY 子句(分组子句)中指定列的值分组，再检索出满足 HAVING 子句中组条件表达式的组，按 SELECT 子句给出的列名或列表达式输出。ORDER 子句(排序子句)用来对输出的目标表进行排序。

- 简单的 SELECT 语句

简单的 SELECT 语句格式如下：

```
SELECT 目标表的列名或列表达式集合
FROM 基本表或(和)视图集合
```

例如：

```
SELECT Sno, Sname, Sgender
```



```
FROM Student
```

这个 SELECT 语句将返回学生表中的选定字段(Sno,Sname,Sgender)的数据。如果需要返回学生表中的所有字段，可以使用星号(*)来代替所有列名，如下语句：

```
SELECT *
FROM Student
```

在 SELECT 语句中可以使用一些常用的数据处理函数，如 AVG(求平均值)、MIN(求最小值)、MAX(求最大值)、SUM(求和)、COUNT(求记录总数)等。

例如，如果要知道学生表 Student 中的记录总数，可以使用下面的 SELECT 语句：

```
SELECT COUNT(*)
FROM Student
```



知识点

在创建表时，为方便计算和查询，一般将字段名称定义为英文名。当需要将表显示出来时，可以使用 AS 关键字来对字段重新命名，例如以下将输出的 Sname 字段重新命名为“姓名”，将字段名称 Sgender 重新命名为“性别”：“SELECT Sname AS 姓名，Sgender AS 性别 FROM Student”。



在 SELECT 语句中数字类型的字段之间还支持加、减、乘和除(+-*/)基本算术操作。而且字符型字段之间也支持加操作，其结果是将两个字符串合并在一起。下面的语句演示了如何把学生表中的学生姓名 Sname 和性别 Sgender 连接起来：

```
SELECT Sname + Sgender AS 学生信息
FROM Student
```

⊙ 使用 WHERE 子句筛选数据

在同样的检索情况下，使用 WHERE 子句可以指定查询条件，缩小数据检索范围，例如要在学生表 Student 中检索出姓名 Sname=“葛萌萌”的记录，可以使用如下 SQL 语句：

```
SELECT *
FROM   Student
WHERE   Sname = '葛萌萌'
```



提示

“葛萌萌”被两个单引号括了起来，这个单引号是为了表示该值是文本(字符串)类型。

当 WHERE 子句的约束条件有多个时，这些约束条件可以通过 AND(而且)、OR(或者)和 NOT(非)逻辑操作符连接起来以实现多个约束。另外，这些约束条件支持小括号运算，在小括号

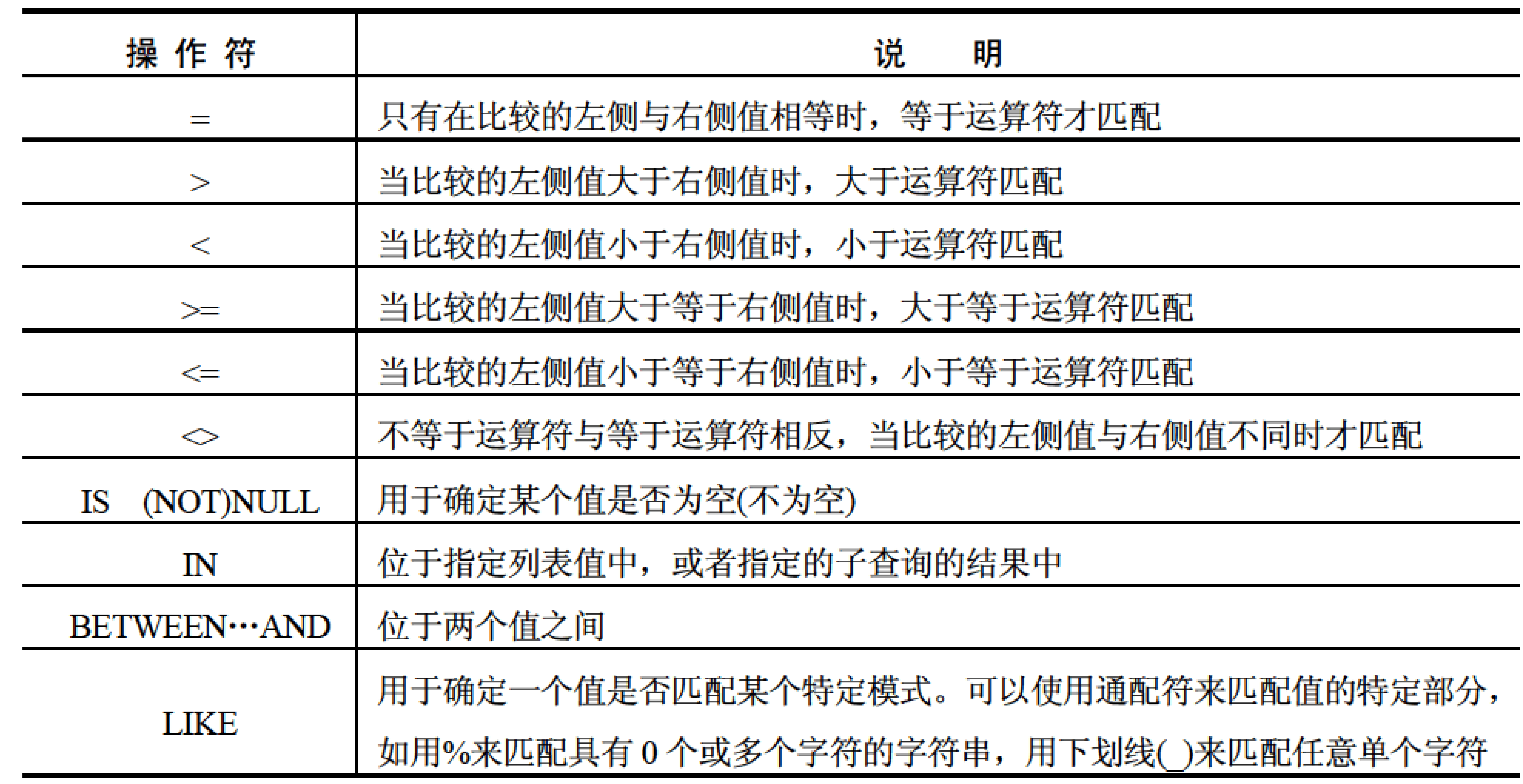
之间的条件将被作为一个整体条件优先执行，例如下面的 SELECT 语句：

```
SELECT *
FROM Student
WHERE   (Cno = 101 OR Sgender = '女')
    AND Sno BETWEEN 10 AND 50
```

执行以上查询语句将返回所有班级号为 101 或性别为“女”且学号在 10~50 之间的学生信息。

在这条 SQL 语句中有一个 BETWEEN…AND 操作符，它表示在某个范围之间。除此之外，在 WHERE 子句还支持很多比较运算符和逻辑运算符，如表 5-5 所示。

表 5-5　WHERE 语句中支持的操作符



操 作 符	说　明
=	只有在比较的左侧与右侧值相等时，等于运算符才匹配
>	当比较的左侧值大于右侧值时，大于运算符匹配
<	当比较的左侧值小于右侧值时，小于运算符匹配
>=	当比较的左侧值大于等于右侧值时，大于等于运算符匹配
<=	当比较的左侧值小于等于右侧值时，小于等于运算符匹配
<>	不等于运算符与等于运算符相反，当比较的左侧值与右侧值不同时才匹配
IS (NOT)NULL	用于确定某个值是否为空(不为空)
IN	位于指定列表值中，或者指定的子查询的结果中
BETWEEN…AND	位于两个值之间
LIKE	用于确定一个值是否匹配某个特定模式。可以使用通配符来匹配值的特定部分，如用%来匹配具有 0 个或多个字符的字符串，用下划线(_)来匹配任意单个字符

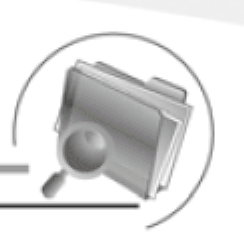
⊙ ORDER BY 子句

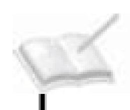
在用 WHERE 子句定义了筛选要求后，如果想改变从数据库返回的数据的顺序。这时可以使用 ORDER BY 子句。ORDER BY 子句出现在 SQL 语句的末尾，可包含一个或多个列名或表达式，也可包括 ASC 或 DESC 来决定记录是以升序(ASC，默认关键字)或以降序(使用 DESC)排列，如果没有指定排序常数，默认将以升序方式排列语法格式如下：

```
ORDER BY  字段或者是字段集合  排序常数
```

例如，将学生表中所有性别为“女”的数据记录，按姓名进行降序排列。

```
SELECT *
FROM Student
WHERE   Sgender = '女'
ORDER BY   Sname   DESC
```




知识点

在 ORDER BY 语句中，即使某个特定的列不是最终结果集的一部分，仍可以以它进行排序。

⦿ 连接查询

连接查询也叫多表查询，在实际应用过程中经常需要同时从两个表或者两个以上的表中检索数据。连接查询允许通过指定表中某个或者某些列作为连接条件，同时从两个表或者多个表中检索数据。

连接查询可以使用两种连接语法形式，一种是把连接条件写在 FROM 子句中，另外一种是把连接条件写在 WHERE 子句中。

连接条件写在 WHERE 子句中的语法形式非常简单，在数据库程序中经常用到，语法格式如下：

```
SELECT  表名.字段名,表名.字段名,...
FROM 表名, 表名…
WHERE  连接条件  AND  搜索条件
```

连接条件一般是表与表之间联系字段的表达式，例如下面的例子：

```
SELECT Student.Sname, Score.degree
FROM Student, Score
WHERE   Student.Sno = Score.Sno AND degree > 60
```


知识点

数据表也可以使用别名，数据表的别名能够大大减少手工输入量，使得代码简洁明了。

连接条件写在 FROM 子句中的形式中需要用到 JOIN 关键字。SQL-92 标准所定义的 FROM 子句的连接语法格式如下：

```
FROM join_table join_type join_table
[ON (join_condition)]
```

其中，join_table 指出参与连接操作的表名，连接可以对同一个表操作，也可以对多个表操作，对同一个表操作的连接又称做自连接；join_type 指出连接类型，可分为内连接、外连接和交叉连接 3 种。

(1) 内连接(INNER JOIN)使用比较运算符进行表间某(些)列数据的比较操作，并列出这些表中与连接条件相匹配的数据行。根据所使用的比较方式不同，内连接又分为等值连接、自然连接和不等连接 3 种。

(2) 外连接分为左外连接(LEFT OUTER JOIN 或 LEFT JOIN)、右外连接(RIGHT OUTER JOIN 或 RIGHT JOIN)和全外连接(FULL OUTER JOIN 或 FULL JOIN)3 种。与内连接不同的是，

外连接不只列出与连接条件相匹配的行，而是列出左表(左外连接时)、右表(右外连接时)或两个表(全外连接时)中所有符合搜索条件的数据行。

(3) 交叉连接(CROSS JOIN)没有 WHERE 子句，它返回连接表中所有数据行的笛卡尔积，其结果集合中的数据行数等于第一个表中符合查询条件的数据行数乘以第二个表中符合查询条件的数据行数。

连接操作中的 ON (join_condition) 子句指出连接条件，它由被连接表中的列和比较运算符、逻辑运算符等构成。


提示

无论哪种连接都不能对 text、ntext 和 image 数据类型进行直接连接，但可以对这 3 种列进行间接连接。

例如：



```
SELECT   Class.Cname, Student.Sname, Student.Saddress, Student.Stelephone, Student.Sgender
FROM     Class INNER JOIN   Student
ON Class.Cno = Student.Cno
```


◉ 联合查询

UNION 运算符可以将两个或两个以上的 SELECT 语句的查询结果集合合并成一个结果集合显示，即执行联合查询。UNION 的语法格式如下：

```
select_statement
UNION [ALL] selectstatement
[UNION [ALL] selectstatement][…n]
```

其中，selectstatement 为待联合的 SELECT 查询语句；ALL 选项表示将所有行合并到结果集合中。不指定该项时，被联合查询结果集合中的重复行将只保留一行。

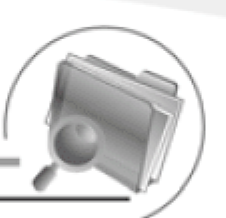
联合查询时，查询结果的列标题为第一个查询语句的列标题。因此，要定义列标题必须在第一个查询语句中定义。要对联合查询结果排序时，也必须使用第一查询语句中的列名、列标题或者列序号。


知识点

在使用 UNION 运算符时，应保证每个联合查询语句的选择列表中有相同数量的表达式，并且每个查询选择表达式应具有相同的数据类型，或是可以自动将它们转换为相同的数据类型。在自动转换时，对于数值类型，系统将低精度的数据类型转换为高精度的数据类型。

3. INSERT 语句

要将新记录插入到表中，可以使用 INSERT 语句。它有一些不同的形式，但最简单的形式如下所示：



```
INSERT INTO 表名
VALUES (第一个字段值,...,最后一个字段值)
```

其中，VALUES 后面的字段值必须与数据表中相应字段所规定的值得数据类型相符，和 WHERE 子句一样，需要将字符串和日期值括在单引号中，但可以在 SQL 语句中直接输入数字和布尔值。如果不想对某些字段赋值，可以用空值 NULL 替代，否则将会产生错误。下面的语句将在学生表中插入一条新的记录：

```
INSERT INTO Student
VALUES(1,'葛萌萌', '女',NULL, '河北省沧州市',13)
```

如果需要插入的是表的某些字段的值，可以在 SQL 语句中使用另一种 INSERT 语句进行操作，其语法格式如下：

```
INSERT INTO 表名(字段 1,…,字段 N,…)
VALUES (第一个字段值,...,第 N 个字段值,…)
```

当用这种形式向数据表中添加新记录时，在关键字 INSERT INTO 后面输入所要添加的数据表名称，然后在括号中列出将要添加新值的字段名称，最后，在关键字 VALUES 的后面按照前面输入的列的顺序对应地输入所有要添加的记录值。需要注意的是，对于表中的非空列，如果没有定义默认值，必须显示地给出该列的值。

4. UPDATE 语句

UPDATE 语句用来修改数据表中已经存在的数据记录。它的基本语法格式如下：

```
UPDATE　表名
SET 字段 1 = 值 1,…, 字段 N = 值 N,
［WHERE 条件表达式］
```

其含义是更新数据表中符合 WHERE 条件的字段或字段集合的值，WHERE 条件是可选的。例如，下面的语句是将学生“葛萌萌”的班级号 Cno 改为 105：

```
UPDATE Student
SET Cno = 105
WHERE Sname = '葛萌萌'
```

上面的 UPDATE 语句是确定值赋值，即为字段赋予指定的值。还可以基于已有的值来设置新的字段值，例如，将成绩表 Score 中课程号为 201 的记录的成绩加 5 分，可以使用如下 SQL 语句：

```
UPDATE　Score
SET degree = degree+5
WHERE　courseID = 201
```

5. DELETE 语句

DELETE 语句用来删除数据表中的记录，基本语法格式如下：

```
DELETE   FROM 表名
[WHERE 条件表达式]
```

与 UPDATE 语句类似，DELETE 语句中的 WHERE 选项也是可选的，如果不限定 WHERE 条件，DELETE 语句将删除数据表中的所有记录，例如：

```
DELETE FROM Student
```

这条语句将删除学生表 Student 中的所有记录。如果并不想清空数据表中所有记录，就必须注意 WHERE 子句的使用。

5.2 ADO.NET 概述



ADO.NET 是.NET Framework 提供的数据访问的类库。ADO.NET 对 Microsoft SQL Server、Oracle 和 XML 等数据源提供一致的访问。应用程序可以使用 ADO.NET 连接到这些数据源，并检索和更新所包含的数据。

5.2.1 ADO.NET 基础

ADO.NET 用于数据访问的类库包含.NET Framework 数据提供程序和 DataSet 两个组件。相应地可以把 ADO.NET 的基本类分为数据提供者对象和数据集对象。提供者对象可用于每一种类型的数据源，专用于提供者的对象完成数据源中实际的读取和写入工作；而数据集对象则是将数据读入到内存中，用来访问和操纵数据。如图 5-2 所示为这些对象之间的关系。

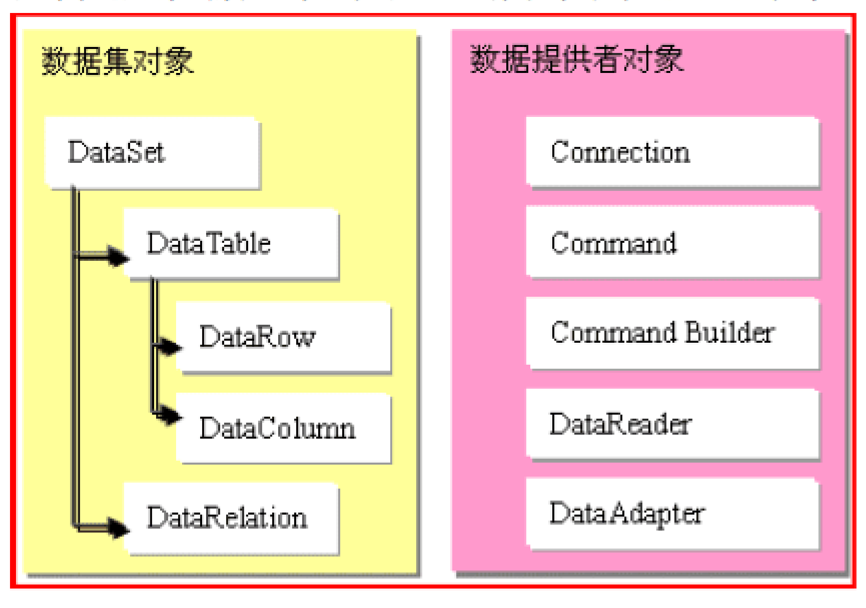
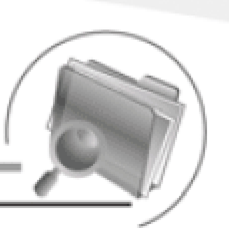


图 5-2　ADO.NET 类之间的关系

提供者对象需要一个活动的连接，可以使用它们先从数据库中读取数据，然后根据需要，通过数据集对象使用内存中的数据，也可以使用提供者对象更新数据源中的数据。数据集对象以非连接方式使用；甚至在数据库连接关闭之后，也可以使用内存中的数据。



1. Data Provider(数据提供程序)

Data Provider 提供了 DataSet 和数据中心(如 MS SQL Server)之间的联系，同时也包含了存取数据中心(数据库)的一系列接口。通过数据提供者所提供的应用程序编程接口(API)，可以轻松地访问各种数据源的数据，包括 OLE DB 和 ODBC 所支持的数据库。

Data Provider 主要有以下 3 部分：

(1) 连接对象 Connection、命令对象 Command、参数对象 Parameter 提供了数据源和 DataSet 之间的接口。

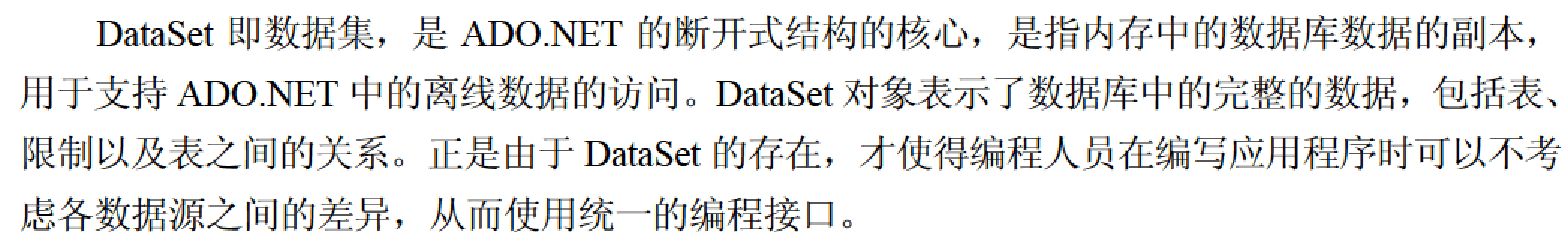
(2) 数据流提供了高性能的、前向的数据存取机制。通过 DataReader 可以轻松而高效地访问数据流，而 DataAdapter 可以用数据源填充 DataSet 并解析更新。

(3) 更底层的对象允许连接到数据库，然后执行数据库系统级的特定命令。

Data Provider 利用本地的 OLEDB 通过 COM Interop 来实现数据存取。OLEDB 支持自动的和手动的事务处理。

2. DataSet





DataSet 即数据集，是 ADO.NET 的断开式结构的核心，是指内存中的数据库数据的副本，用于支持 ADO.NET 中的离线数据的访问。DataSet 对象表示了数据库中的完整的数据，包括表、限制以及表之间的关系。正是由于 DataSet 的存在，才使得编程人员在编写应用程序时可以不考虑各数据源之间的差异，从而使用统一的编程接口。

运行时，组件可以交换数据集。也就是说，一个组件可以将数据集传递给另一个组件。为了适应在组件间进行数据集交换，ADO.NET 使用了一个基于 XML 的保持和传递格式。ADO.NET 解决方案将内存中的数据(数据库)表示为一个 XML 文件，然后将这个 XML 文件发送给另一个组件。

用户可以使用 DataSet 对象对数据集中的内容进行处理。DataSet 对象允许使用与关系模型一致的方法对数据集的内容进行处理。例如，DataSet 对象有一个 DataTable 对象集合，每个 DataTable 对象都有行和列，并且与其他的 DataTable 对象有关联。当一个组件将数据集传递给另一个组件时，接收组件将把接收到的数据集物化为一个 DataSet 对象。

5.2.2　提供者对象

提供者对象就是指在每一个.NET 数据提供者中定义的对象，其名称前带有特定提供者的名称。因此，用于 OLE DB 提供者的连接对象就是 OleDbConnection；用于 SQL Server .NET 提供者的类就是 SqlConnection。

在 ADO.NET 中，连接数据源有 4 种接口：SQLClient、OracleClient、ODBC、OLEDB。其中 SQLClient 是 Microsoft SQL Server 数据库专用连接接口，OracleClient 是 Oracle 数据库专用的连接接口，ODBC 和 OLEDB 可用于其他数据源的连接。在应用程序中使用任何一种连接接口时，必须在后台代码中引用对应的命名空间，类的名称也随之发生变化，如表 5-6 所示。

表 5-6　数据连接方式命名空间与对应的类名称

名 称 空 间	对应的类名称
System.Data.SqlClient	SqlConnection、SqlCommand、SqlDataReader、SqlDataAdapter
System.Data.Odbc	OdbcConnection、OdbcCommand、OdbcDataReader、OdbcDataAdapter
System.Data.OleDb	OleDbConnection、OleDbCommand、OleDbDataReader、OleDbDataAdapter
System.Data.OracleClientt	OracleConnection、OracleCommand、OracleDataReader、OracleDataAdapter

1. 连接对象

连接对象是使用 ADO.NET 访问数据库的第一个对象，它提供了到数据源的基本连接。如果使用的数据库需要用户名和密码，或者是位于远程网络服务器上，则连接对象可以提供建立连接并登录的细节。根据数据源的不同，连接对象有 4 种：SqlConnection、OleDbConnection、OdbcConnection 和 OracleConnection。

连接对象的常用属性和方法如表 5-7 所示。


表 5-7　连接对象的常用属性和方法

属性或方法	描　述
ConnectionString	该属性用来指定连接的数据源，需要使用很多参数：如 Data Source 指明数据源；Initial Catalog 指明数据库；Integrated Security 指明集成安全；User ID 和 Password 分别用于指明登录帐户和密码等
ConnectionTimeout	该属性用于获取在尝试建立连接时终止尝试并生成错误之前所等待的时间，单位为秒，默认值为 15
Database	该属性返回当前数据库的名称或连接打开后要使用的数据库名称，默认为空字符串
DataSource	获取要连接的数据源实例的名称
Open	该方法用于打开由 ConnectionString 属性指定的数据源连接
Close	该方法用于断开由 ConnectionString 属性指定的数据源连接

2. 命令对象

命令对象用于向数据源发出命令。命令对象可直接执行 SQL 语句或存储过程，其 CommandText 属性就是要执行的 SQL 语句，如“SELECT * FROM Customers”。对于不同的提供者，该对象的名称也略有不同：例如，用于 SQL Server 的命令对象为 SqlCommand，用于 ODBC 的为 OdbcCommand，用于 OLE DB 的命令对象为 OleDbCommand，用于 Oracle 的命令对象为 OracleCommand。

Command 对象的常用属性和方法如表 5-8 所示。

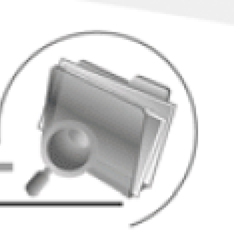


表 5-8　Command 对象的常用属性和方法

属性或方法	描　述
CommandText	获取或设置要对数据源执行的 SQL 语句或存储过程
CommandTimeout	获取或设置在终止执行命令的尝试并生成错误之前的等待时间，以秒为单位，默认值为 30
CommandType	获取或设置 CommandText 的类型，其值为 System.Data.CommandType 值之一，默认为 Text
Connection	获取或设置 Command 实例使用的 Connection 对象
ExcuteNonQuery	该方法对连接的数据库执行 SQL 语句并返回影响的行数，返回值为受影响的行数
ExcuteReader	将 CommandText 发送到 Connection 并生成一个 DataReader，返回值为一个 DataReader 对象
ExcuteScalar	执行查询并返回查询所返回的结果集中的第一行第一列，忽略其他行或列
ExcuteXmlReader	将 CommandText 发送到 Connection 并生成一个 System.Xml.XmlReader 对象，返回值为该 XmlReader 对象
ResetCommandTimeout	将 CommandTimeout 属性重置为默认值

3. CommandBuilder 对象

此对象用于构建 SQL 命令，在基于单一表查询的对象中进行数据修改。对于不同的提供者，该对象的名称分别为：用于 SQL Server 的 SqlCommandBuilder，用于 ODBC 的 OdbcCommandBuilder，用于 OLE DB 的 OleDbCommandBuilder 和用于 Oracle 的 OracleCommandBuilder。

CommandBuilder 对象的常用属性和方法如表 5-9 所示。

表 5-9　CommandBuilder 对象的常用属性和方法

属性或方法	描　述
DataAdapter	获取或设置自动为其生成 SQL 语句的一个 DataAdapter 对象
GetUpdateCommand	获取自动生成的、对数据库执行更新操作所需的 Command 对象
GetDeleteCommand	获取自动生成的、对数据库执行删除操作所需的 Command 对象
GetInsertComman	获取自动生成的、对数据库执行插入操作所需的 Command 对象

4. DataReader 对象

该对象用于从数据源中读取仅能前向和只读的数据流。对于简单的数据读取来说，此对象的性能最好。对于不同的提供者，该对象的名称分别为：用于 SQL Server 的 SqlDataReader，用于 ODBC 的 OdbcDataReader、用于 OLE DB 的 OleDbDataReader 和用于 Oracle 的 OracleDataReader。

DataReader 对象的常用属性和方法如表 5-10 所示。

表 5-10　DataReader 对象的常用属性和方法

属性或方法	描　　述
FieldCount	获取当前行中的列数
RecordsAffected	被更改、插入或删除的行数
IsClosed	指示是否可关闭数据读取器
Close	关闭 DataReader 对象
GetName	获取指定列的名称
Read	使 DataReader 前进到下一条记录，如果存在多个行则返回 true，否则返回 false
NextResult	当读取批处理 SQL 语句的结果时，使数据读取器前进到下一个结果，如果存在多个结果集则返回 true，否则返回 false
IsDBNull	获取一个值，指示列中是否包含不存在的或已丢失的值
GetOrdinal	在给定列名称的情况下获取列序号

5. DataAdapter 对象

DataAdapter(数据适配器)是 DataSet 和数据源之间的桥梁，可以执行针对数据源的各种操作，包括更新变动的数据，填充 DataSet 对象以及其他操作。对于不同的提供者，该对象的名称分别为：用于 SQL Server 的 SqlDataAdapter，用于 ODBC 的 OdbcDataAdapter、用于 OLE DB 的 OleDbAdapter 和用于 Oracle 的 OracleDataAdapter。

在创建 DataAdapter 对象时，可以直接指定 Connection 和 Command 对象。如果要定义后指定属性，主要包括：SelectCommand、InsertCommand、UpdateCommand 和 DeleteCommand。

DataAdapter 对象的常用方法主要有以下 3 个。

- Fill：该方法用来执行 SelectCommand，用数据源的数据填充 DataSet 对象。
- GetData：该方法新建一个数据集中 DataTable 并填充它。
- Update：更新数据集中的某个 DataTable。

5.2.3　数据集对象

数据集对象位于 System.Data 命名空间中，用于定义 ADO.NET 的断开的、客户端的对象，包括 DataSet、DataTable、DataRow、DataColumn 和 DataRelation 等。

1. DataSet 对象

DataSet 是数据集对象中的首要对象，此对象表示一组相关表，在应用程序中这些表作为一个单元来引用。例如，Student、Class 和 Score 是一个 DataSet 中的 3 张表，有了此对象，就可以快速从每一个表中获取所需要的数据，当与服务器断开时检查并修改数据，然后在另一个操作中使用这些修改的数据更新服务器。



DataSet 允许访问低级对象，这些对象代表单独的表和关系。这些对象是 DataTable 对象和 DataRelation 对象。

DataSet 对象的常用方法如表 5-11 所示。

表 5-11　DataSet 对象的常用方法

方　法	描　述
AcceptChanges	提交自加载此 DataSet 或上次调用 AcceptChanges 以来对其进行的所有更改
BeginInit	开始初始化在窗体上使用或由另一个组件使用的 DataSet，初始化发生在运行时
EndInit	结束在窗体上使用或由另一个组件使用的 DataSet 的初始化
Clear	通过移除所有表中的所有行来清除任何数据的 DataSet
Clone	复制 DataSet 的结构，包括所有 DataTable 架构、关系和约束。不复制任何数据
Copy	复制 DataSet 的结构和数据，返回新的 DataSet，具有与该 DataSet 相同的结构和数据
Merge	将指定的 DataSet 及其架构合并到当前 DataSet 中
GetChanges	获取 DataSet 的副本，该副本包含自加载以来或上次调用 AcceptChanges 方法以来所有的更改，可以对该副本执行操作



2. DataTable 对象

该对象代表 DataSet 中的一个表。例如，Student。

DataTable 对象的 Rows 和 Columns 分别是 DataRow 和 DataColumn 对象，可用于访问 DataTable 表中的行和列。含义分别如下。

- DataColumn 对象：代表表中的一列，如 Sno 或 Sname。
- DataRow 对象：代表来自表的关联数据的一行；例如 Student 表中的 Sno、Sname 和 Saddress 等。

3. DataRelation 对象

该对象代表通过共享列而发生关系的两个表之间的关系；例如 Student 表中的 Cno 列标识学生所在的班级。于是，可以创建 DataRelation 对象，通过共享 Cno 列建立 Student 和 Class 表之间的关系。

5.2.4　使用 ADO.NET 访问数据库

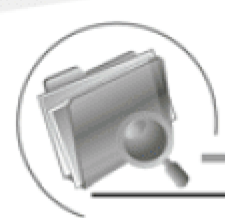
ASP.NET 数据访问程序的开发流程有以下几个步骤：

(1) 利用 Connection 对象创建数据连接。

(2) 利用 Command 对象数据源执行 SQL 命令。

(3) 利用 DataReader 对象读取数据源的数据。

(4) DataSet 对象与 DataAdapter 对象配合，完成数据的查询和更新操作。



1. 使用数据库资源管理器

通过 VWD 的【数据库资源管理器】可以直接操纵 SQL Server 数据库，但需要在该面板的【数据连接】下进行注册。下面以注册数据库 InfoManage 为例介绍如何添加数据连接。

(1) 选择【视图】|【其他窗口】|【数据库资源管理器】命令，打开【数据库资源管理器】面板，默认情况下，该面板与【解决方案资源管理器】面板在一起。

(2) 在【数据库资源管理器】面板中右击【数据连接】，从弹出的快捷菜单中选择【添加连接】命令，打开【选择数据源】对话框，选择【Microsoft SQL Server】选项，如图 5-3 所示。

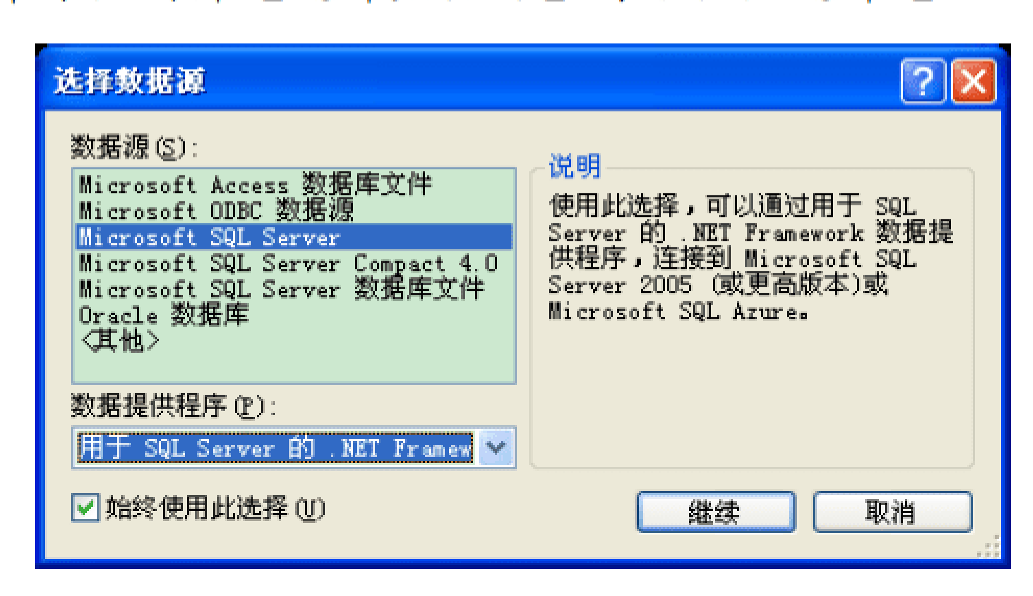


图 5-3 【选择数据源】对话框



知识点

【选择数据源】对话框只有在第一次添加连接时才弹出，下一次添加数据连接时，默认还为 SQL Server，如果要添加其他类型的数据源，可以单击【添加连接】对话框中的【更改】按钮。

(3) 单击【继续】按钮，打开【添加连接】对话框，在【服务器名】文本框中输入 SQL Server 服务的名称，也可以单击【刷新】按钮，然后从下拉列表中选择，根据服务器的设置选择登录设置，然后单击【测试连接】按钮，如图 5-4 所示，如果连接成功将弹出如图 5-5 所示的测试连接成功对话框。

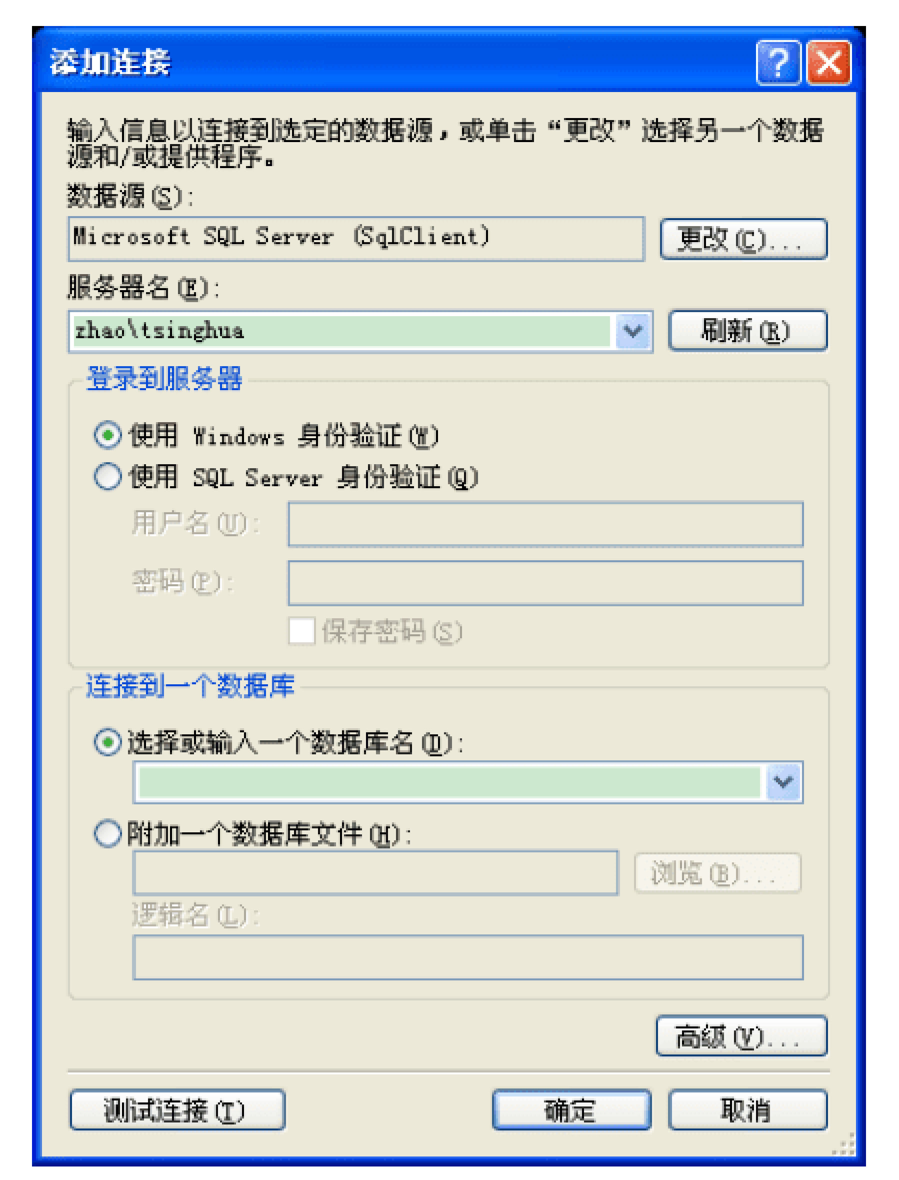


图 5-4 【添加连接】对话框



图 5-5 测试连接成功对话框





(4) 连接成功后即可从【选择或输入一个数据库】下拉列表中选择要连接的数据库，如图 5-6 所示。

(5) 单击【确定】按钮关闭【添加连接】对话框，此时的【数据库资源管理器】面板中将增加添加的数据库，如图 5-7 所示。

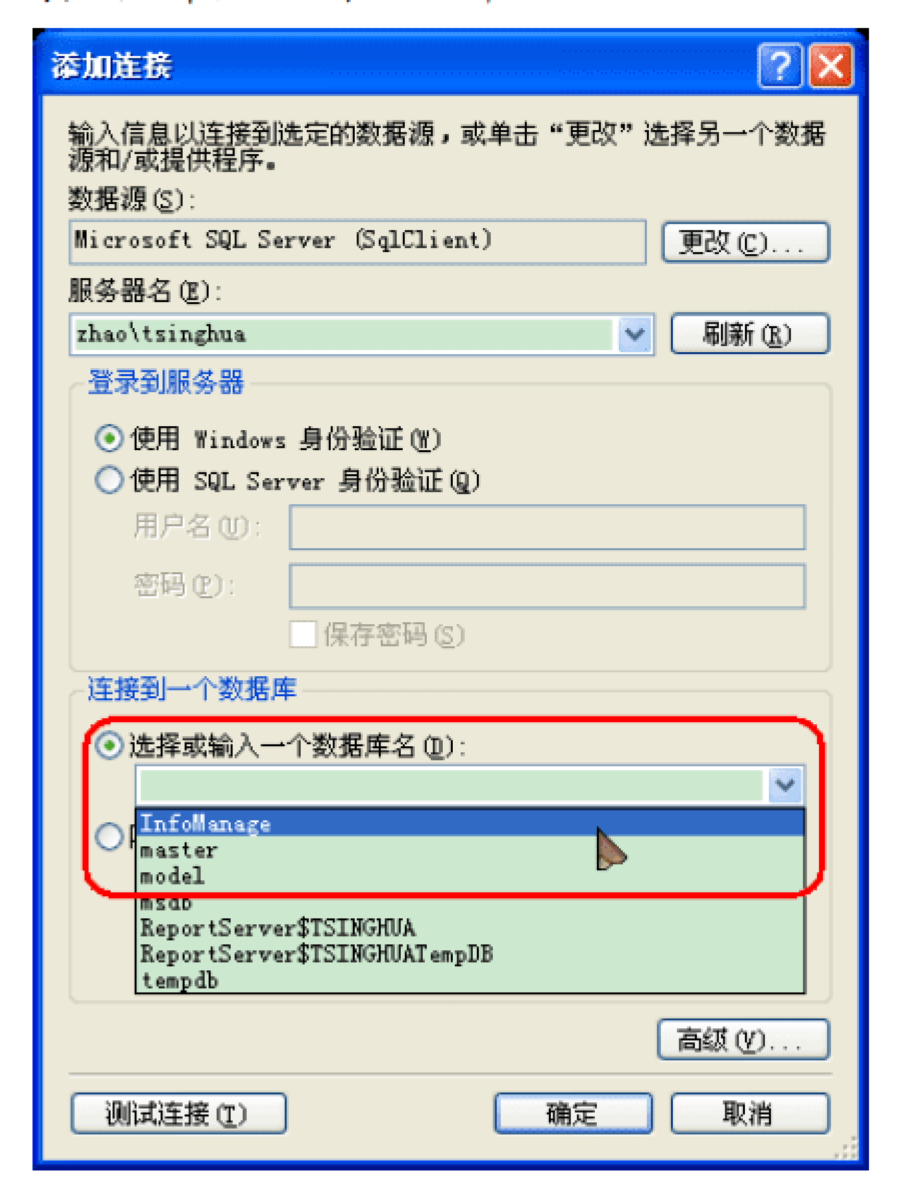



图 5-6　选择数据库

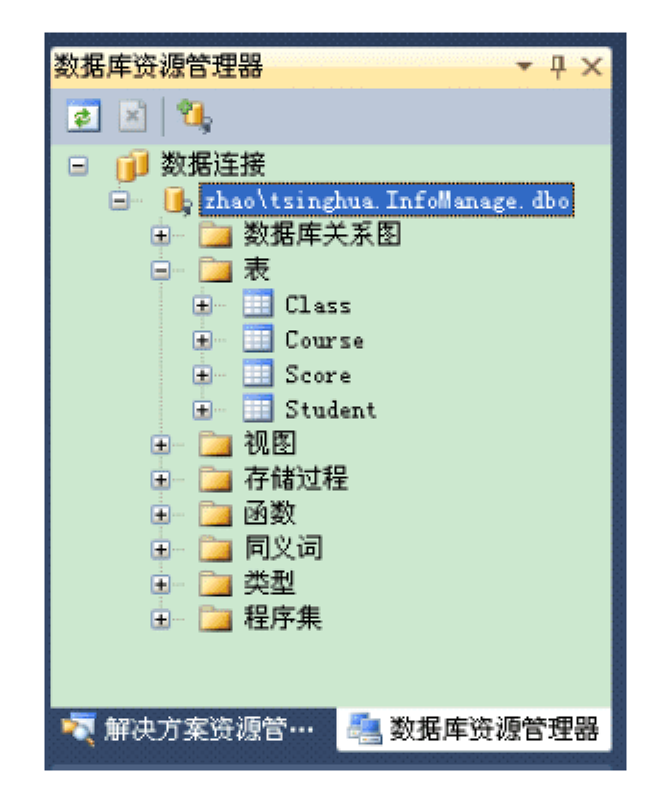



图 5-7　添加连接后的【数据库资源管理器】

此时，就可以在 VWD 中对 InfoManage 数据库进行任何设计、查询和修改操作，就像在 SQL Server Management Studio 一样。


提示

有关数据库的操作请读者参考相关书籍，本书不做详细介绍。

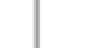
2. 学生信息查询与维护

为了让读者更深刻地理解数据提供者对象和数据集对象的用法，下面做一个具体的实例。本例功能比较简单，支持学生信息的添加、查询和删除功能。

【例 5-1】通过 ADO.NET 访问 SQL Server 数据库 InfoManage，实现对 Student 表的添加、查询和删除功能。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 5-1】。

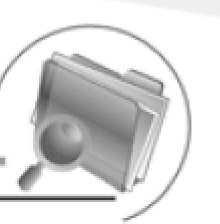
(2) 在 Default.aspx 页面的设计视图中，添加 1 个 RadioButtonList 控件和 3 个 Panel 控件。为 RadioButtonList 控件添加 3 个 Item 元素，分别对应 3 个 Panel 控件，用于控制 Panel 控件的显示与隐藏，生成的代码如下：



```
<asp:RadioButtonList ID="RadioButtonList2" runat="server" AutoPostBack="True"
    onselectedindexchanged="RadioButtonList2_SelectedIndexChanged"
    RepeatDirection="Horizontal">
    <asp:ListItem Selected="True" Value="Panel1">添加学生信息</asp:ListItem>
    <asp:ListItem Value="Panel2">查询学生信息</asp:ListItem>
    <asp:ListItem Value="Panel3">删除学生信息</asp:ListItem>
</asp:RadioButtonList>
```

(3) 在 Panel1 控件中插入一个 8 行 2 列的表格，该表格用于布局输入学生信息的控件。将第一行的 2 个单元格合并，后面的第一列输入文本信息，第二列添加相应的控件，最终生成的代码如下：

```
<asp:Panel ID="Panel1" runat="server">
    <table class="style1">
        <tr>
            <td colspan="2" style="text-align: center">
                添加学生——输入学生信息</td>
        </tr>
        <tr>
            <td>
                学号：</td>
            <td>
                <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                    ControlToValidate="TextBox1" ErrorMessage="学号不能为空">
                </asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td>
                姓名：</td>
            <td>
                <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
                    ControlToValidate="TextBox2" ErrorMessage="请输入学生姓名">
                </asp:RequiredFieldValidator>
            </td>
        </tr>
        <tr>
            <td>
                性别：</td>
```



```
                <td>
                    <asp:RadioButtonList ID="RadioButtonList1" runat="server"
                        RepeatDirection="Horizontal">
                        <asp:ListItem Selected="True" Value="M">男</asp:ListItem>
                        <asp:ListItem Value="F">女</asp:ListItem>
                    </asp:RadioButtonList>
                </td>
            </tr>
            <tr>
                <td>
                    联系电话：</td>
                <td>
                    <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td>
                    地址：</td>
                <td>
                    <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td>
                    班级：</td>
                <td>
                    <asp:DropDownList ID="DropDownList1" runat="server">
                    </asp:DropDownList>
                </td>
            </tr>
            <tr>
                <td style="text-align: right">
                    <asp:Button ID="Button1" runat="server" style="text-align: right" Text="提交"
                        onclick="Button1_Click" />
                </td>
                <td>
                     </td>
            </tr>
        </table>
    </asp:Panel>
```





(4) Panel2 用于查询学生信息，添加相应的控件，生成如下代码:

```
<asp:Panel ID="Panel2" runat="server" Visible="False">
按学号查询学生信息<br />
学号：<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
    <asp:Button ID="Button2" runat="server" Text="查询" onclick="Button2_Click" />
<hr />
    <asp:Label ID="Label1" runat="server" Text="学生信息："></asp:Label>
</asp:Panel>
```

(5) Panel3 用于删除学生信息，添加相应的控件，生成如下代码:

```
<asp:Panel ID="Panel3" runat="server" Visible="False">
删除学生信息<br />
学号：<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
    <asp:Button ID="Button3" runat="server" Text="删除" onclick="Button3_Click" />
</asp:Panel>
```



(6) 由于要访问数据库，所以在后台代码文件中需要引入相应的命名空间:

```
using System.Data.SqlClient;
using System.Data;
```

(7) 定义两个类成员变量，分别用于存放连接字符串和班级信息，其中 Dictionary<K,V>用于定义键/值(Key/Value)对的集合，需要引入 System.Collections 命名空间，代码如下:

```
    string strConnect = "Data Source=zhao\\tsinghua;Initial Catalog=InfoManage;Integrated
Security=True;user=sa;password=Sapassword";
    Dictionary<int, string> classInfo = new Dictionary<int, string>();
```

(8) 因为学生表中存放的是班级号，而展现给用户的需要班级名称，所以需要查询班级表 Class 对应班级号和班级名称。在页面的 Load 事件中，加载班级信息，添加如下代码:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        getClassInfo();
    }
}
private void getClassInfo()
{
    DropDownList1.Items.Clear();
    SqlConnection con = new SqlConnection(strConnect);
```

```
        con.Open();
        SqlCommand cmd = new SqlCommand("select * from Class", con);
        SqlDataReader reader = cmd.ExecuteReader();
        while (reader.Read())//逐行遍历查询结果
        {//用班级信息初始化“班级”下拉列表
            int cno = reader.GetInt32(0);
            string cname = reader.GetString(1);
            ListItem item = new ListItem(cname, cno.ToString());
            DropDownList1.Items.Add(item);
            classInfo.Add(cno, cname);
        }
        cmd = null;
        con.Close();
        con = null;
    }
```

(9) 单选按钮列表框控制 Panel 控件的显示与隐藏，相应的事件处理程序如下：

```
    protected void RadioButtonList2_SelectedIndexChanged(object sender, EventArgs e)
    {
        Panel1.Visible = false;
        Panel2.Visible = false;
        Panel3.Visible = false;
        if (RadioButtonList2.SelectedValue == "Panel1")
            Panel1.Visible = true;
        if (RadioButtonList2.SelectedValue == "Panel2")
            Panel2.Visible = true;
        if (RadioButtonList2.SelectedValue == "Panel3")
            Panel3.Visible = true;
    }
```

(10) 前面查询班级信息使用的是命令对象和 DataReader 对象，下面的添加和查询操作改用 DataAdapter 和数据集的方式实现，相应的代码如下：

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlConnection con = new SqlConnection(strConnect);
        DataSet ds = new DataSet();
        string info = "";
        try
        {
            con.Open();
```

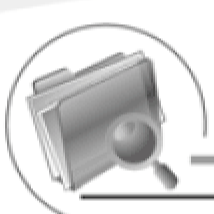
```
            SqlDataAdapter sqld = new SqlDataAdapter("select * from student", con);
            //建立 CommandBuilder 对象来自动生成 DataAdapter 的 Command 命令，否则就要自己编写
            //Insertcommand ,deletecommand , updatecommand  命令。
            SqlCommandBuilder cb = new SqlCommandBuilder(sqld);
            sqld.Fill(ds, "student");//用 Fill 方法填充 DataSet
            DataTable dTable = ds.Tables["student"];//将数据表的数据复制到 DataTable 对象
            DataRow row = ds.Tables["student"].NewRow();//增加新记录
            //给该记录赋值
            row[0] = TextBox1.Text;
            row[1] = TextBox2.Text;
            row[2] = RadioButtonList1.SelectedValue;
            row[3] = TextBox3.Text;
            row[4] = TextBox4.Text;
            row[5] = DropDownList1.SelectedValue;
            ds.Tables["student"].Rows.Add(row);
            sqld.Update(ds, "student");//提交更新
            info = "添加学生信息成功";
        }
        catch (Exception ex)
        {
            info = "添加失败，失败原因：" + ex.Message;
        }
        finally
        {
            con.Close();
            con = null;
        }
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\""+ info +"\"); ", true);
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        if (TextBox5.Text == "")
        {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"请输入要查询学生的
学号\"); ", true);
            return;
        }
        SqlConnection con = new SqlConnection(strConnect);
        DataSet ds = new DataSet();
        con.Open();
```



```
        SqlDataAdapter sqld = new SqlDataAdapter("select * from Student where Sno = @SNO", con);
        sqld.SelectCommand.Parameters.AddWithValue("@SNO",TextBox5.Text);
        sqld.Fill(ds, "student");//用 Fill 方法填充 DataSet
        DataTable dTable = ds.Tables["student"];//将数据表的数据复制到 DataTable 对象
        DataRowCollection rows = dTable.Rows;//获取数据行
        //逐行遍历，取出各行的数据
        Label1.Text = "学生信息：";
        if (rows.Count > 0)
            getClassInfo();
        else
        {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"该学生的信息不存
在\"); ", true);
            return;
        }
        for (int i = 0; i < rows.Count; i++)
        {
            DataRow row = rows[i];
            Label1.Text += "<br />学号：" + row[0];
            Label1.Text += "；    姓名：" + row[1];
            if(row[2].ToString()=="M")
                Label1.Text += "；    性别：男";
            else
                Label1.Text += "；    性别：女";
            Label1.Text += "；    联系电话：" + row[3];
            Label1.Text += "；    地址：" + row[4];
            foreach(KeyValuePair<int,string> kv in classInfo )
            {
                if(kv.Key == Int32.Parse(row[5].ToString()))
                {
                    Label1.Text += "；    班级：" + kv.Value;
                    break;
                }
            }
        }
        con.Close();
        con = null;
    }
```

(11) 对于删除操作，与增加和查询类似，既可以采用命令对象的方式，也可以使用





DataAdapter，这里使用 Comand 对象来删除学生记录，代码如下：

```
protected void Button3_Click(object sender, EventArgs e)
{
    if (TextBox6.Text == "")
    {
        Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\"请输入要删除学生的
学号\"); ", true);
        return;
    }
    SqlConnection con = new SqlConnection(strConnect);
    SqlCommand sqlcommand = new SqlCommand("delete from Student where Sno=@no",con);
    sqlcommand.Parameters.AddWithValue("@no", TextBox6.Text);
    string info = "";
    try
    {
        con.Open();
        int DeleteCount = sqlcommand.ExecuteNonQuery();
        if ( DeleteCount > 0)
            info = "成功删除记录";
        else
            info = "该学生信息不存在";
    }
    catch (Exception ex)
    {
        info = "删除失败，错误原因：" + ex.Message;
    }
    finally
    {
        sqlcommand = null;
        con.Close();
        con = null;
    }
    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "success", "alert(\""+ info + "\"); ", true);
}
```

(12) 编译并运行程序，在添加学生信息页面，【班级】下拉列表中动态加载了当前所有的班级，如图 5-8 所示。输入相应的信息，单击【提交】按钮将弹出添加成功对话框，如图 5-9 所示。如果输入信息有误，将添加失败并给出错误信息，如图 5-10 所示。

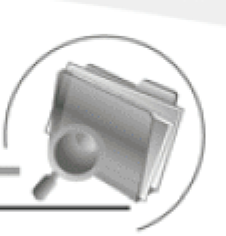




图 5-8　添加学生



图 5-9　添加成功

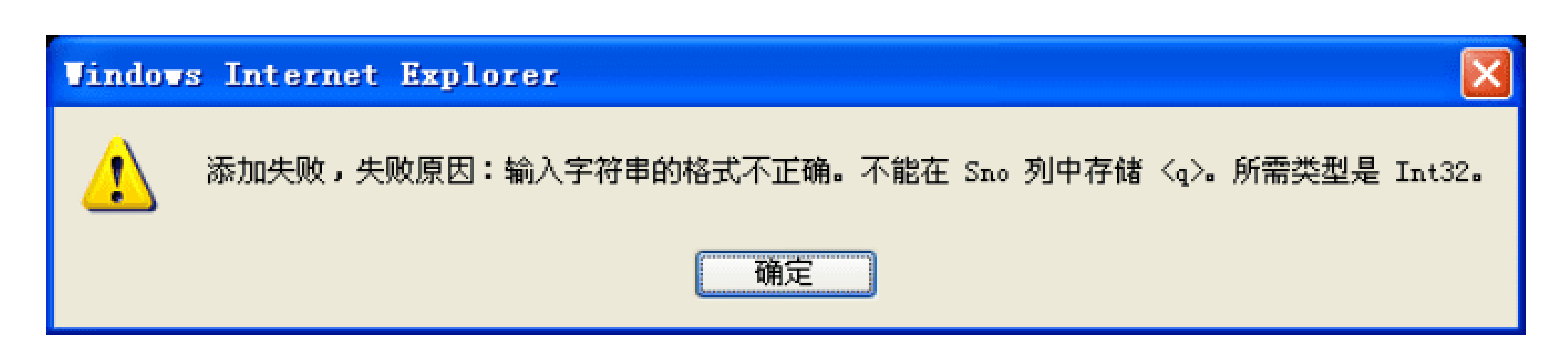


图 5-10　添加失败

(13) 选中【查询学生信息】单选按钮，输入学号，单击【查询】按钮，将显示相应的学生信息，如图 5-11 所示。如果没有找到，则弹出对话框提示学生不存在，如图 5-12 所示。

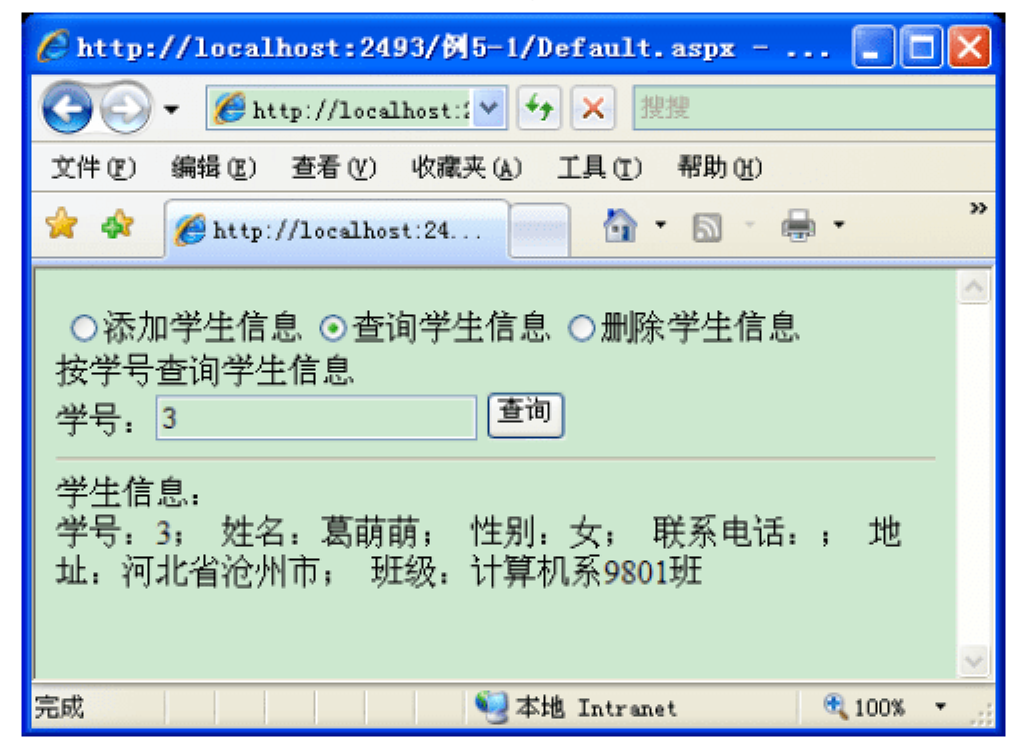


图 5-11　查询学生信息

图 5-12　学生不存在

(14) 删除操作也类似，请读者实际操作，观察效果。



5.3 数据绑定和数据控件

前面已经介绍了通过 ADO.NET 访问数据库，下面将继续介绍如何利用 ASP.NET 提供的控件将数据呈现在页面上。首先介绍单值绑定和列表控件的数据绑定过程，然后介绍 GridView 等复杂数据绑定控件的基本用法，主要涉及以下 3 种复杂数据绑定控件：GridView、DataList 和 FormView。

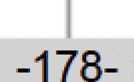
5.3.1 数据绑定概述

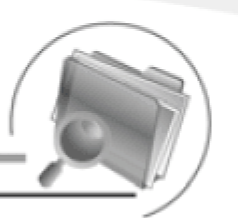
Web 系统的一个典型的特征是后台对数据的访问和处理与前台数据的显示分离，而前台显示是通过 HTML 来实现的。一种将数据呈现的最直接的方式是将需要显示的数据和 HTML 标记拼接成字符串并输出，但这种方案的缺点也是显而易见的，不但复杂而且难以重用，尤其是有大量数据需要处理时。为了简化开发过程，ASP.NET 环境中提供了多种不同的服务器端控件来帮助程序员更快速高效地完成数据的呈现。这些用于数据呈现的 ASP.NET 控件，集成了常见的数据显示框架和数据处理功能，因而在使用时只需要设置某些属性，并将需要显示的数据交付给控件，控件就可以帮助用户按照固定的样式(例如表格)或通过模板自定义样式将一系列数据呈现出来，同时还自动继承某些内置的数据处理功能，例如：排序、分页等。这些控件就被称为数据绑定控件，而将数据交付给数据绑定控件的过程就称为数据绑定。

数据绑定控件本质上依然是通过 HTML 来呈现数据的，只不过按照某种样式生成 HTML 框架并将数据填入其中的工作由控件自动完成了，一些复杂的数据绑定控件还提供了大量的功能帮助用户对数据进行进一步操作，例如：排序、过滤、新增、修改和删除等，因而使得数据呈现的过程变得简单而灵活。正因为如此，数据绑定控件的使用是 ASP.NET 编程中非常重要的一部分内容。

5.3.2 单值和列表控件的数据绑定

数据绑定实际上是在 HTML 标记中或服务器控件中设置要显示数据的过程。对于页面中的 HTML 标记，可以直接嵌入数据或绑定表达式来设置要显示的数据，而对于服务器控件来说，通常通过设置控件属性或指定数据源来完成数据的绑定，并控制其呈现的样式。常用的绑定表达式具有以下形式：<%# XXX%>，绑定表达式可以直接嵌入到前台页面代码中去，通常用于 HTML 标记中的数据显示或单值控件数据设置，例如 Label、TextBox 等。而对于列表控件(如：DropDownList、CheckBoxList)，以及后面要着重介绍的复杂数据绑定控件则常采用设置数据源的方式完成数据呈现。





1. 单值绑定

单值绑定其实就是实现动态文本的一种方式。为了实现单值绑定，可影响页面中添加一些特殊的绑定表达式。主要有以下 4 种数据绑定表达式。

- <%= XXX %>：内联引用方式，可以引用 C#代码。
- <%# XXX %>：可以引用.cs 文件中的字段，但该字段必须初始化后，在页面的 Load 事件中使用 Page.DataBind 方法来实现。
- <%#$ XXX %>：可以引用 web.config 文件中预定义的字段或者已注册的类。
- <%# Eval(XXX) %>：类似于 JavaScript，数据源也需要绑定。

例如，下面的数据绑定表达式都是有效的：

```
<%# DateTime.Now %>
<%# 3+(6*number) %>   //其中，number 是 Web 也后置代码类中的 public 或 protected 变量
<%# Request.Browser.Browser %>
```



提示

单值绑定有两个缺点：(1) 数据绑定代码与定义用户界面的代码混合在一起；(2) 代码过于分散。正是由于这两个缺点，因此不方便管理页面和代码，所以应尽量少用单值绑定。



2. 多值绑定

多值绑定通常与列表控件以及复杂的数据控件一起工作，可以把多条数据一次绑定到这些控件中。

多值绑定的步骤如下：

(1) 把存储数据的数据对象绑定到列表控件或数据控件的 DataSource 属性；

(2) 调用控件或者当前页面的 DataBind()方法。

为了创建多值绑定，需要使用支持数据绑定的控件，ASP.NET 提供了一系列这类控件，这些控件如下。

- 列表控件：ListBox、DropDownList、CheckBoxList 和 RadioButtonList 等。
- HtmlSelect 控件：这是一个 HTML 控件，类似于 ListBox 控件。
- 复杂数据控件：GridView、DetailsView、FormView、ListView 等。

【例 5-1】中 DropDownList 控件就可以通过数据绑定来进行初始化操作，请读者在学完数据源控件以后进行修改。

5.3.3 数据控件简介

为了有效地处理系统中的数据，ASP.NET 工具箱的【数据】类别中提供了两组数据感知控



件：数据源控件和数据绑定控件。数据源控件用于从数据源(如数据库或 XML 文件)中检索数据，然后将这一数据提供给数据绑定控件；数据绑定控件可用于显示和编辑数据。

1. 数据源控件

数据源控件用于连接数据源、从数据源中读取数据以及把数据写入数据源。数据源控件不呈现任何用户界面，而是充当特定数据源与 ASP.NET 网页上的其他控件之间的桥梁。数据源控件实现了各种数据检索和修改功能，其中包括查询、排序、分页、筛选、更新、删除以及插入等。

在工具箱的【数据】类别下包含了 7 个不同的数据源控件。

- ObjectDataSource 控件：具有数据检索和更新功能的中间层对象，允许使用业务对象或其他类，并可创建依赖中间层对象管理数据的 Web 应用程序。
- SqlDataSource 控件：用来访问存储在关系数据库中的数据源，包括 Microsoft SQL Server、Oracle、OLE DB 和 ODBC。当该控件与 SQL Server 一起使用时支持高级缓存功能；当数据源作为 DataSet 对象返回时，该控件还支持排序、筛选和分页功能。
- AccessDataSource 控件：用于在 Web 页面中显示 Microsoft Access 数据库的数据。它非常简单，在某种程度上类似于 SqlDataSource 控件，因为它允许处理来自数据库的数据。但不同之处在于它只针对 Microsoft Access 数据库进行优化。
- XmlDataSource 控件：主要用于绑定分层的、基于 XML 的数据，该控件支持使用 XPath 表达式来实现筛选功能，并允许对数据应用 XSLT 转换，此外，还允许通过保存更改后的 XML 文档来更新数据。
- SiteMapDataSource 控件：该控件结合导航控件实现站点导航功能，第 3 章在介绍导航控件时曾介绍过如何使用 SiteMapDataSource。
- EntityDataSource 控件：EntityDataSource 控件之于 EF(Entity Framework)就像 SqlDataSource 控件之于基于 SQL 的数据源，它提供了一个声明性的方法来访问模型。和 SqlDataSource 控件一样，EntityDataSource 提供了对 CRUD 操作的轻松访问，另外使数据排序和筛选也变得非常简单。EntityDataSource 通过 LINQ to EF 提供了对底层 SQL Server 数据库的完全访问。
- LinqDataSource 控件：用作 LINQ to SQL 的数据源。LINQ to SQL 是一种类似于 EF(将在第 6 章介绍)的技术。因为 Microsoft 现在大力推广的是 EF，而不是 LINQ to SQL，所以本书不讨论该控件。
- QueryExtender 控件：用作 LinqDataSource 和 EntityDataSource 控件的增件，因为它可以用来创建丰富的过滤界面，从而能够搜索特定的数据，而不必手动编写大量代码。

本章重点介绍 SqlDataSource 控件，SqlDataSource 控件使用 ADO.NET 类与 ADO.NET 支持的任何数据库进行交互。通过该控件，可以在 ASP.NET 页面中访问和操作数据，而无需直接使用 ADO.NET 类，只需提供用于连接到数据库的连接字符串，并定义使用数据的 SQL 语句和存储过程即可。在运行时，SqlDataSource 控件会自动打开数据库连接，执行 SQL 语句或存储过程，返回指定数据，然后关闭连接。





下面通过一个具体的实例来学习 SqlDataSource 控件的使用。

【例 5-2】使用数据源控件和数据绑定初始化【例 5-1】中的 DropDownList 控件。

(1) 启动 VWD 2010，打开网站【例 5-1】。

(2) 切换到 Default.aspx 的设计视图，在工具箱的【数据】类别中双击 SqlDataSource 控件，将添加该控件并打开控件的【任务】面板，单击【配置数据源】选项，如图 5-13 所示。

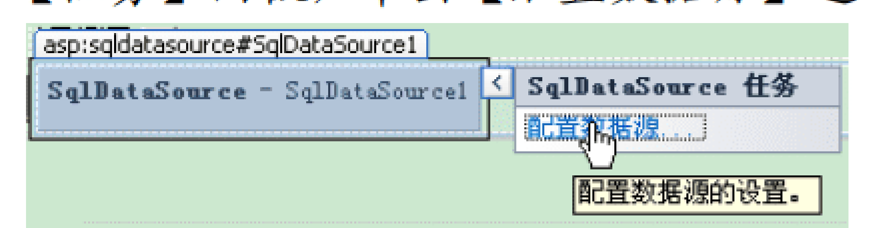


图 5-13　SqlDataSource 控件的【任务】面板

(3) 打开【配置数据源】对话框，第一步是选择数据连接，因为前面已经添加了数据连接，所以在此可以在下拉列表中直接选择，如果下拉列表中没有，可以单击【新建连接】按钮添加数据连接。单击【连接字符串】前面的加号，可以查看该连接生成的连接字符串，如图 5-14 所示。

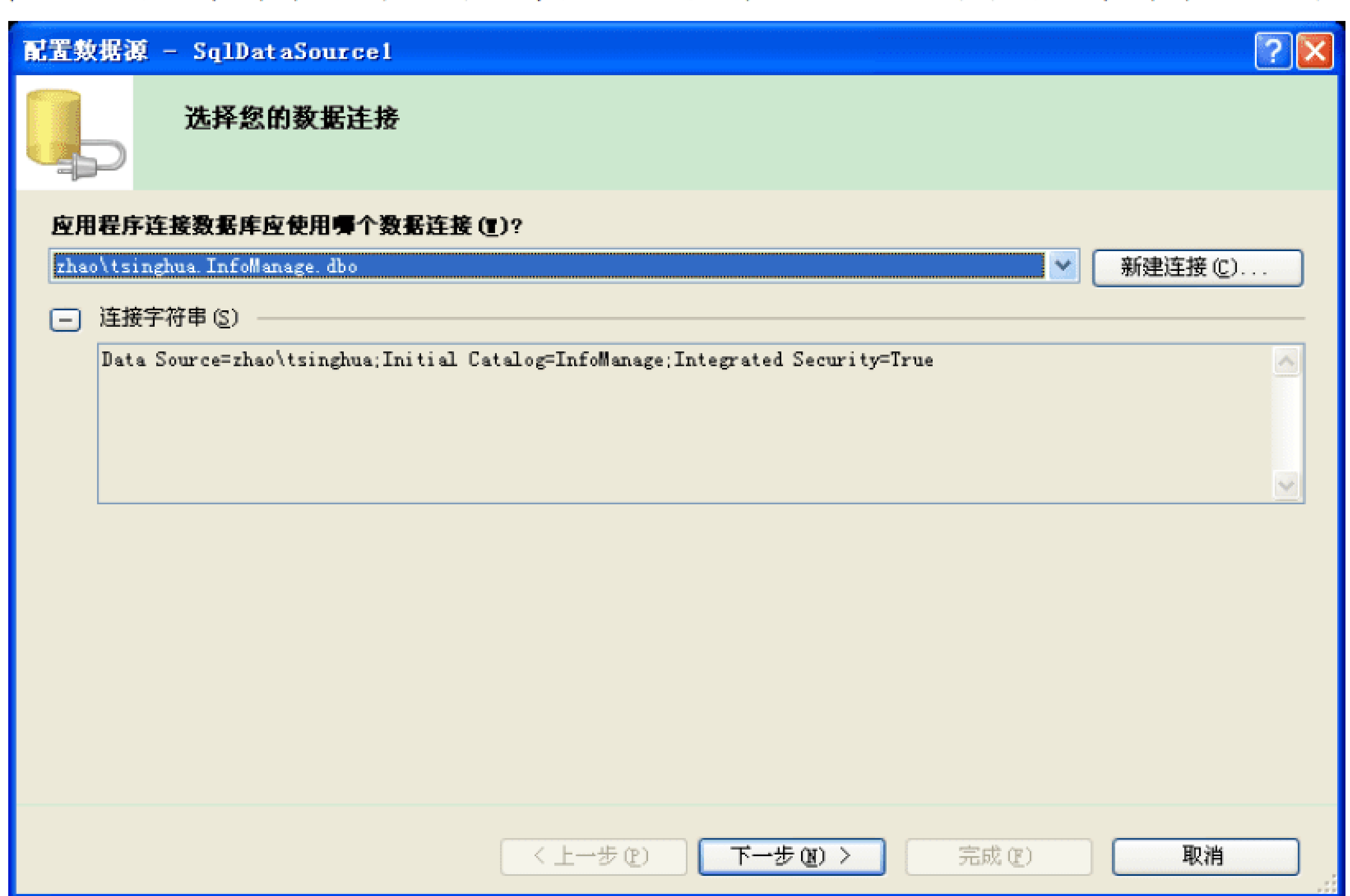


图 5-14　选择数据连接

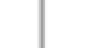


知识点

在 web.config 文件中的<connectionStrings>元素下将创建上述连接字符串，然后数据源控件将通过下面的语句访问这个连接字符串：ConnectionString="<%$ ConnectionStrings:InfoManageConnectionString %>"。

(4) 单击【下一步】按钮，打开如图 5-15 所示的【将连接字符串保存到应用程序配置文件中】对话框。

计算机基础与实训教材系列



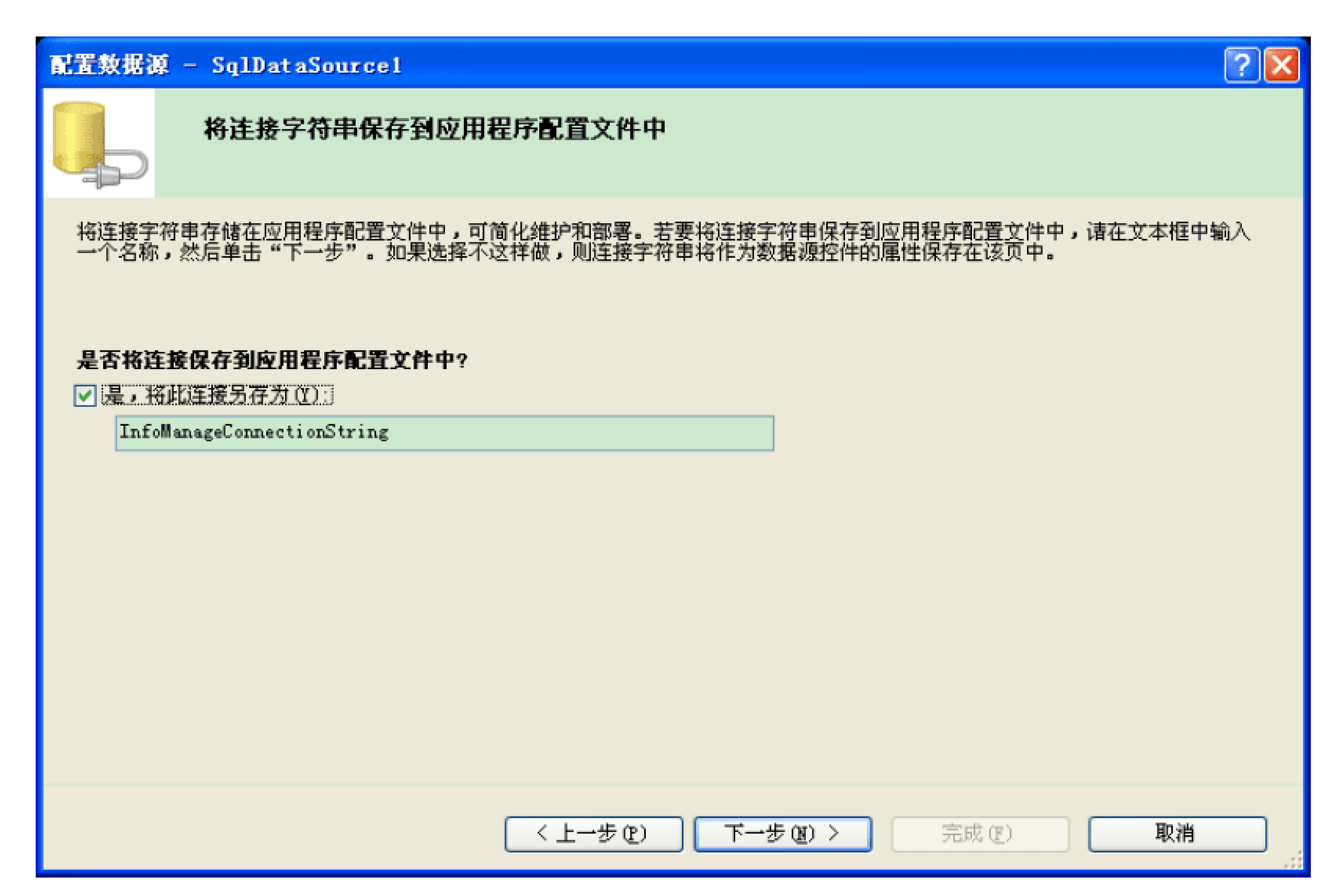


图 5-15　【将连接字符串保存到应用程序配置文件中】对话框

(5) 选中【是，将此连接另存为】复选框，单击【下一步】按钮，打开如图 5-16 所示的【配置 Select 语句】对话框。在该对话框中可以选择需要的表和列，也可以增加 WHERE 子句和 ORDER BY 子句，还可以单击【高级】按钮进行高级 SQL 生成选项设置。

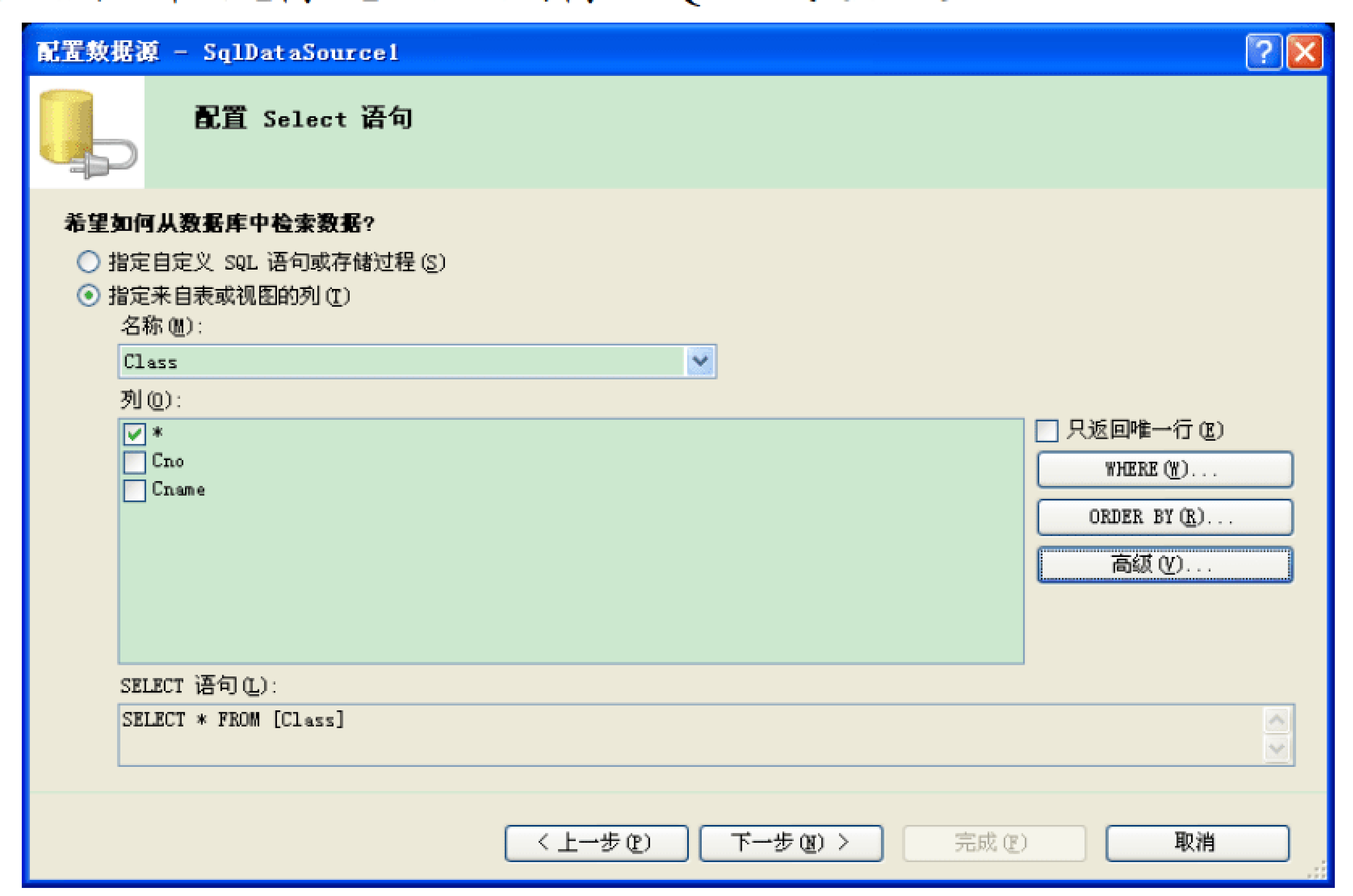


图 5-16　【配置 Select 语句】对话框

(6) 单击【下一步】按钮，打开如图 5-17 所示的【测试查询】对话框，单击【测试查询】按钮，将显示上一步配置的 Select 语句的查询结果。



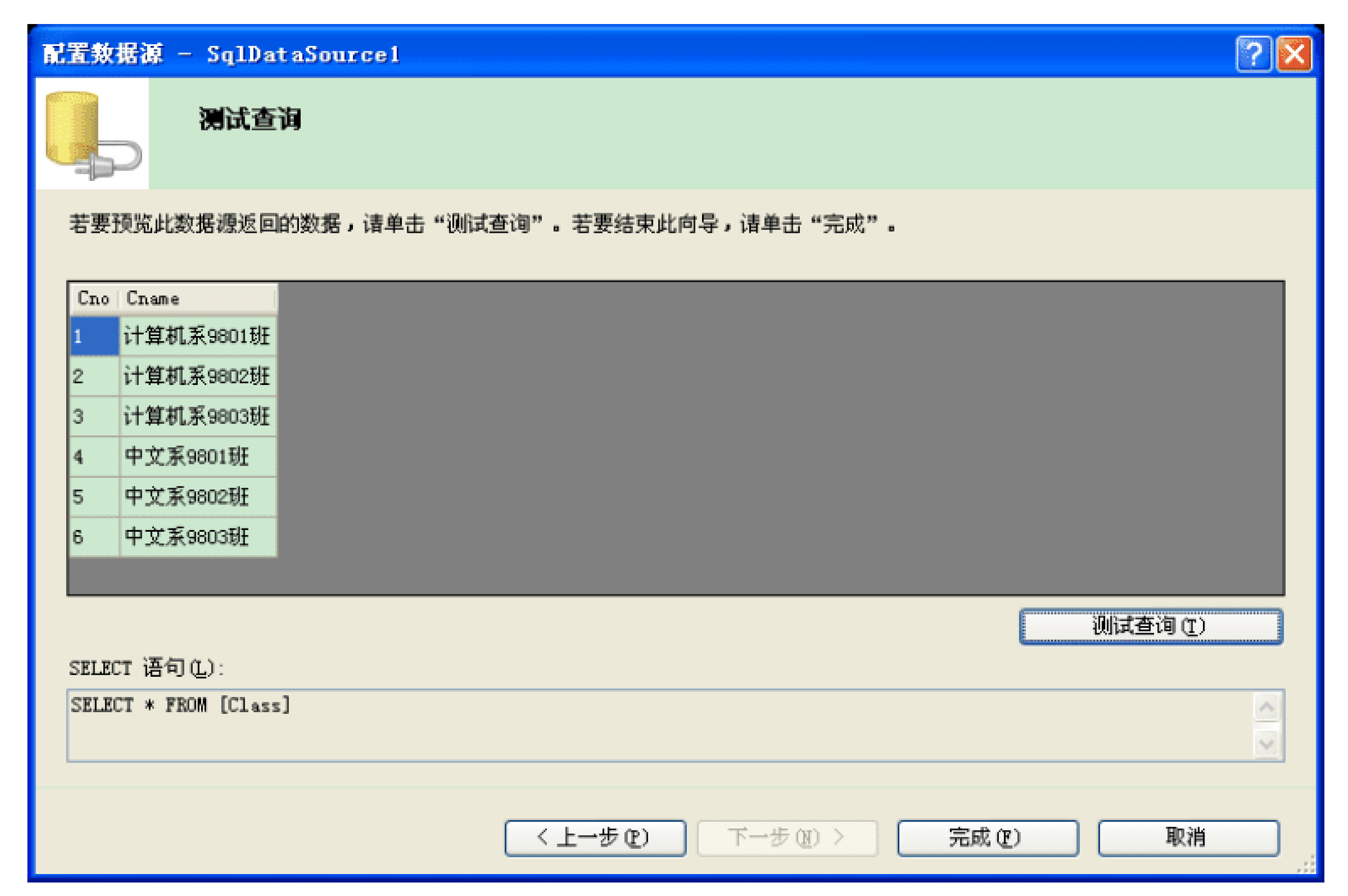


图 5-17　【测试查询】对话框

(7) 单击【完成】按钮，结束数据源的配置。

(8) 在【属性】面板中设置 DropDownList 控件的数据绑定相关的属性，包括 DataSourceID、DataMember、DataTextField 和 DataValueField，如图 5-18 所示。

(9) 此时可以删除【例 5-1】中初始化 DropDownList 控件的代码，然后编译并运行程序。

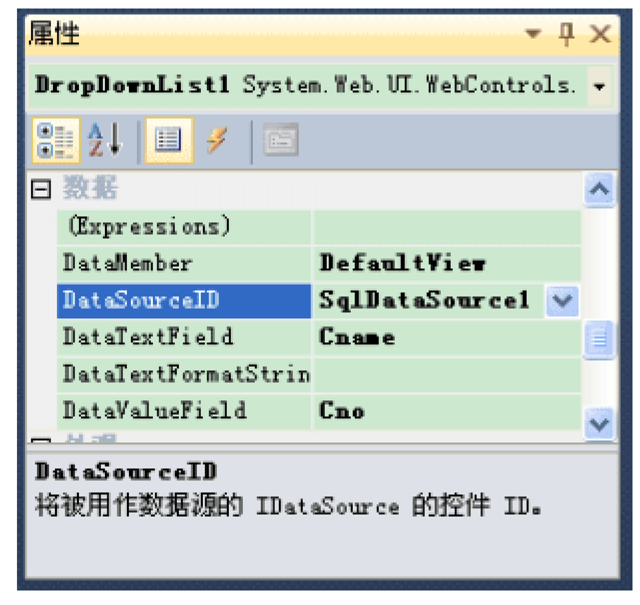


图 5-18　设置数据绑定相关的属性

知识点

数据源控件不呈现任何用户界面，所以添加数据控件时，无论在设计视图的任何位置，最终运行时都没有影响。



2. 数据绑定控件

使用数据绑定控件可以在 Web 页面上显示和编辑数据。在 VWD 2010 工具箱的【数据】类别中有 7 个数据绑定控件。其中，GridView、DataList、ListView 和 Repeater 都可以同时显示多条记录，而 DetailsView 和 FormView 设计为一次显示一条记录，DataPager 是为 ListView 控件提供分页功能的辅助控件。

- GridView 控件：这是一个功能非常多的控件，它支持自动分页(记录被划分到多个“页面”中)、排序、编辑、删除和选择。它像一个带有行和列的电子表格那样呈现数据，其中每行包含一条完整的记录。尽管有许多种可以样式化这些行和控件外观的方法，但不能从根本上改变表现数据的方式。另外，GridView 并不允许直接在底层数据源中插

入记录。

- DataList 控件：该控件不仅可以像 GridView 那样以行表现数据记录，也可以以列的形式表现，从而可以创建一种矩阵形式的数据表现方法。另外，它也允许通过一组模板定义数据的外观。不过，它不支持分页和排序，不允许插入新记录或更新、删除已有记录。
- Repeater 控件：该控件在输出到浏览器的 HTML 方面提供了最大的灵活性，因为该控件本身并不添加任何 HTML 到页面输出中。因此，它常用于 HTML 有序列表或无序列表(<ol>和<ul>)，以及其他列表形式。可以通过控件提供的大量模板定义整个客户端标记。不过，该控件没有分页、排序和修改数据的内置功能。
- ListView 控件：该控件是在 ASP.NET 3.5 中引入的，它很好地合并了 GridView、DataList 和 Repeater 的功能。ASP.NET 4.0 中对 ListView 做出了一些改动，使它更易于使用。类似于 GridView，它支持数据编辑、删除和分页；像 DataList 那样，它支持多列和多行布局，而且还像 Repeater 那样允许完全控制控件生成的标记。
- DetailsView 控件：和 FormView 控件有些类似，它们都只能一次显示一条记录。DetailsView 使用内置的表格格式显示数据，而 FormView 使用模板来定义数据的外观。
- FormView 控件：ASP.NET 4.0 中对 FormView 添加了一个新的 RenderOuterTable 属性。当把这个属性设为 True(默认值为 False)时，控件不会生成包装的 HTML <table>元素，这样就会生成更少的代码和更清晰的 HTML。
- DataPager 控件：该控件可以在其他控件上分页，目前它只能用于扩展 ListView 控件，但随着.NET Framework 未来版本的发布，这一情况将会改观。



3. 其他数据控件

工具箱中【数据】类别中还有一个控件是 Chart 控件。它最初是作为 Visual Studio 2008 的一个增件发布的，但是现在已经完全集成到了 Visual Studio 2010 中。它可用来绘制从简单的条形图到 3D 饼图和折线图的各种图形。这个控件不在本书的讨论范围之内，所以这里不再介绍。

5.3.4 使用数据控件

本节将使用 GridView、FormView 和 DetailsView 控件以主-从表的形式显示成绩表、学生表和课程表中的数据。

【例 5-3】使用数据绑定控件实现主-从数据显示。

(1) 启动 VWD 2010，新建网站【例 5-3】。

(2) 切换到 Default.aspx 的设计视图，添加 GridView、FormView 和 DetailsView 控件各一个。

(3) 为 GridView 控件添加数据源。为该控件新建数据源，生成的数据源代码如下所示：

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:InfoManageConnectionString %>"
    SelectCommand="SELECT * FROM [Score]"></asp:SqlDataSource>
```

(4) 在【GridView 任务】面板中选择【编辑列】选项，打开【字段】对话框，可以在该对话框中设置 GridView 控件显示的字段。在【选定的字段】列表中，将每个字段的 HeaderText 属性都改为中文，通过增加显示命令按钮列，如图 5-19 所示。

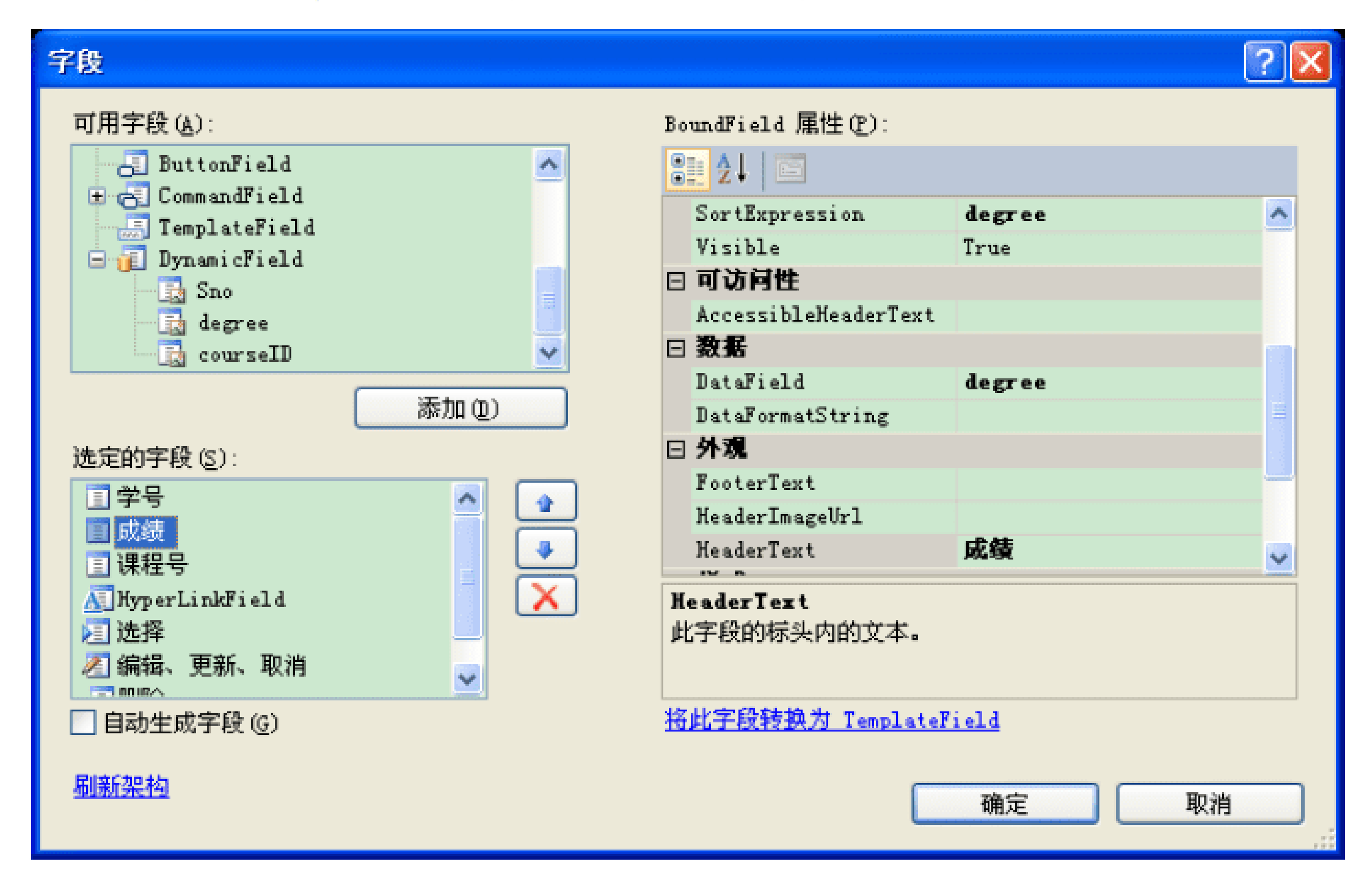


图 5-19　【字段】对话框

(5) 选中【GridView 任务】面板中的【启用分页】和【启用排序】复选框，在【属性】面板中设置控件的 DataKeyNames 属性为“Sno，courseID”。

(6) 通过【GridView 任务】面板中的【自动套用格式】选项设置控件的外观。

(7) 为 FormView 控件新建数据源，在【配置 Select 语句】一步中选项 Student 表，然后单击【WHERE】按钮打开【添加 WHERE 子句】对话框，设置【列】、【运算符】、【源】和【控件 ID】分别为“Sno”、“=”、“Control”和“GridView”，如图 5-20 所示。单击【高级】按钮，在弹出的对话框中选中【生成 INSERT、UPDATE 和 DELETE 语句】复选框，这样就可以启用 FormView 控件的插入、更新和删除功能了。生成的数据源代码如下：

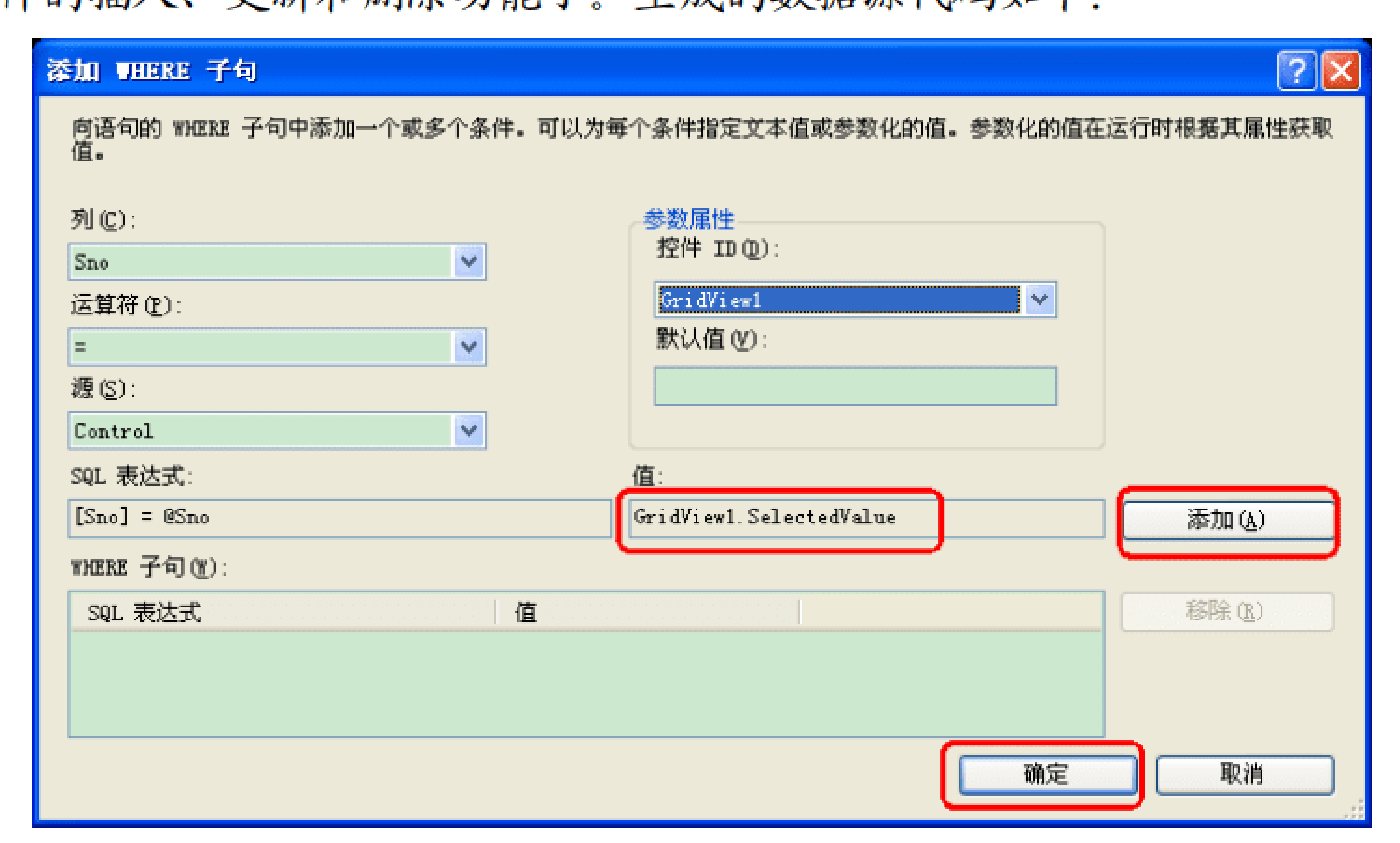
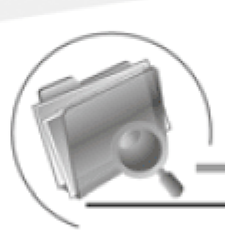


图 5-20　【添加 WHERE 子句】对话框






```
<asp:SqlDataSource ID="SqlDataSource2" runat="server"
    ConnectionString="<%$ ConnectionStrings:InfoManageConnectionString %>"
    SelectCommand="SELECT * FROM [Student] WHERE ([Sno] = @Sno)"
    ConflictDetection="CompareAllValues"
    <SelectParameters>
        <asp:ControlParameter ControlID="GridView1" Name="Sno"
            PropertyName="SelectedValue]" Type="Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

(8) 上述代码中 GridView 控件的 SelectedValue 属性对应 GridView 控件的 DataKeyNames 字段的值，由于前面设置 GridView 控件的 DataKeyNames 为两个字段，所以此处略作修改，将<SelectParameters>修改为如下代码:



```
<SelectParameters>
     <asp:ControlParameter ControlID="GridView1" Name="Sno"
         PropertyName="SelectedDataKey.Values[0]" Type="Int32" />
</SelectParameters>
```


(9) 通过【FormView 任务】面板中的【编辑模板】选项，将控件中的字段信息都修改为中文，通过【自动套用格式】选项设置控件的外观。

(10) 用类似的方法为 DetailsView 控件添加数据源，同样需要修改<SelectParameters>参数，相应的数据源代码如下:

```
<asp:SqlDataSource ID="SqlDataSource3" runat="server"
    ConnectionString="<%$ ConnectionStrings:InfoManageConnectionString %>"
    SelectCommand="SELECT * FROM [Course] WHERE ([courseID] = @courseID)">
    <SelectParameters>
        <asp:ControlParameter ControlID="GridView1" Name="courseID"
            PropertyName="SelectedDataKey.Values[1]" Type="Int32" />
    </SelectParameters>
</asp:SqlDataSource>
```

(11) 同样也对 DetailsView 控件进行编辑模板和自动套用格式设置。

(12) 至此已完成所有工作，没有编写任何代码，一个简单的主-从数据显示页面就做好了。编译并运行程序，查看效果，如果 5-21 所示。

(13) 当选择 GridView 控件中的某一条记录时，下面将显示该行记录对应的学生信息和课程信息。同时可以对成绩表中的数据进行编辑和删除操作，可以对学生信息进行编辑、删除和新建操作。



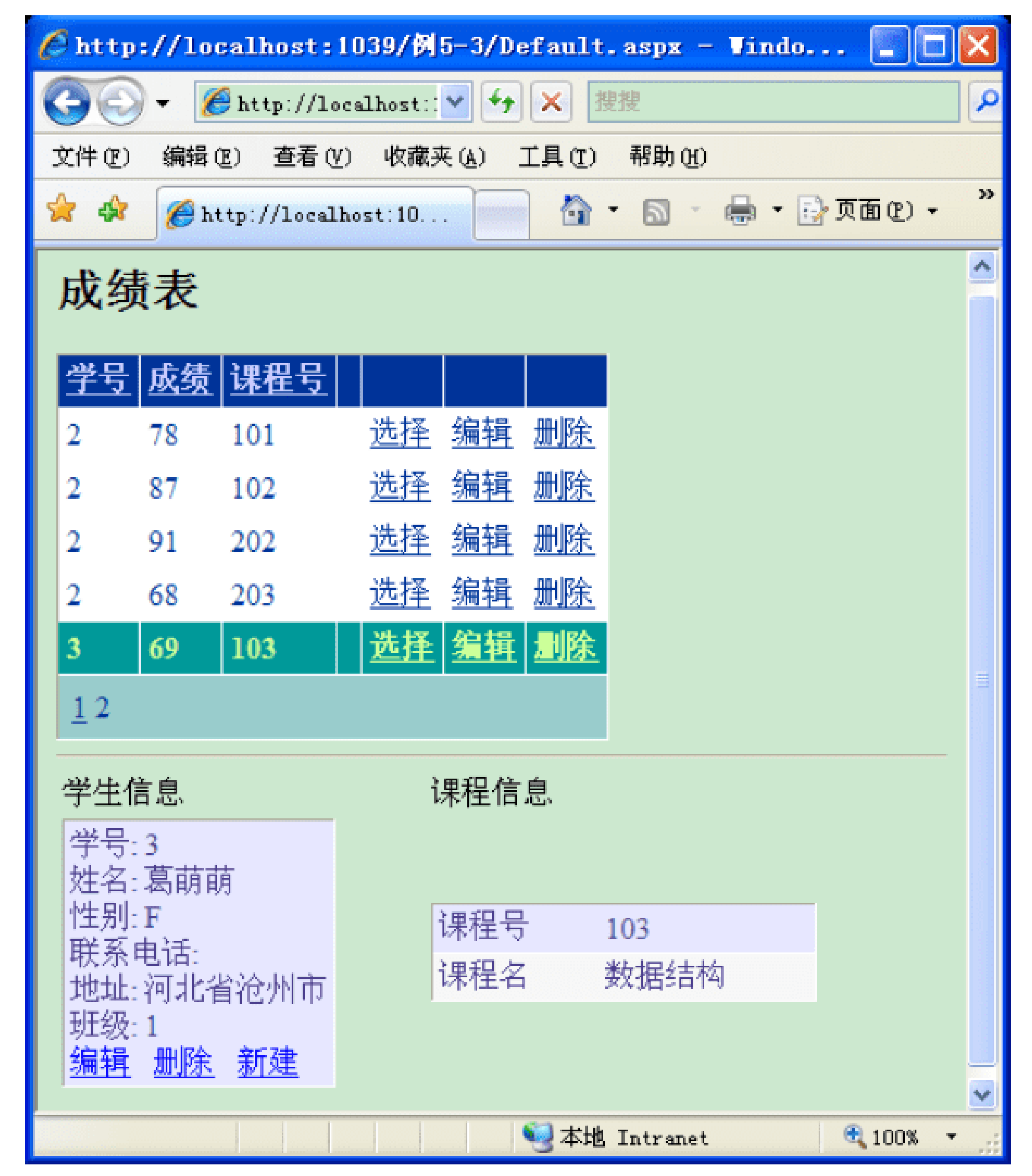


图 5-21　主-从信息表显示

5.4　上机练习

本章的上机练习主要介绍数据库中的事务处理和 DataList 控件的使用。

5.4.1　事务处理

对于数据库管理系统来说，如果没有显式定义事务的开始和结束，就默认一条 SQL 语句为一个单独事务，多数情况下采用这种默认方式就足够了。但是，有时需要将一组 SQL 语句作为一个事务，要么全部执行成功，要么全部不执行。

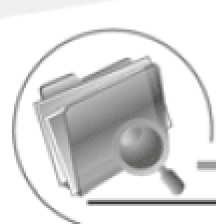
在 ASP.NET 中，可以使用 Connection 和 Transaction 对象开始、提交和回滚事务。一般步骤如下：

(1) 调用 Connection 对象的 BeginTransaction 方法来标记事务的开始，BeginTransaction 方法返回对 Transaction 的引用；

(2) 将 Transaction 对象赋给 Command 的 Transaction 属性；

(3) 执行事务操作；

(4) 如果事务操作成功，使用 Transaction 对象的 Commit 方法提交事务，否则，使用 Rollback



方法回滚事务。

下面来看一个具体的实例。对学生表中姓名为“葛萌萌”的学生班级号改为 4；将班级号为 4 的班级名称为“经管学院 1101 班”。

(1) 启动 VWD 2010，新建网站【上机练习 5-1】。

(2) 向 Default.aspx 页面中添加一个 Label 控件，一个 Button 控件，设置 Button 控件的 Text 属性为“事务提交”。

(3) 为按钮控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    string strConnect = "Data Source=zhao\\tsinghua;Initial Catalog=InfoManage;Integrated
Security=True;user=sa;password=Sapassword";
    SqlConnection conn = new SqlConnection(strConnect);
    conn.Open();
    //开始事务
    SqlTransaction tran = conn.BeginTransaction();
    SqlCommand sqlcommand = new SqlCommand("update Student set Cno=4 where Sname = '葛萌萌'",conn);
    sqlcommand.Connection = conn;
    sqlcommand.Transaction = tran;
    try
    {
        sqlcommand.ExecuteNonQuery();
        sqlcommand.CommandText = "update Class set Cname='经管学院 1101 班' where Cno=4";
        sqlcommand.ExecuteNonQuery();
        tran.Commit();
        Label1.Text = "事务提交成功";
    }
    catch (Exception ex)
    {
        tran.Rollback();
        Label1.Text = "事务提交失败：" + ex.Message;
    }
    finally
    {
        sqlcommand = null;
        conn.Close();
        conn = null;
    }
}
```





(4) 编译并运行程序。

5.4.2　DataList 的数据绑定

在 DataList 控件中可以实现对关系数据集的编辑、更新、插入、删除和分页等数据处理功能。DataList 控件中通过自定义模板设置数据的显示样式，它支持如下模板类型。

- ItemTemplate：包含一些 HTML 元素和控件，将为数据源中的每一行呈现一次这些 HTML 元素和控件。
- AlternatingItemTemplate：包含一些 HTML 元素和控件，将为数据源中的每两行呈现一次这些 HTML 元素和控件。通常可以使用此模板来为交替行创建不同的外观，例如指定一个与在 ItemTemplate 属性中指定的颜色不同的背景色。
- SelectedItemTemplate：包含一些元素，当用户选择 DataList 控件中的某一项时将呈现这些元素。通常可以使用此模板来通过不同的背景色或字体颜色直观地区分选定的行。还可以通过显示数据源中的其他字段来展开该项。
- EditItemTemplate：指定当某项处于编辑模式中时的布局。此模板通常包含一些编辑控件，如 TextBox 控件。
- HeaderTemplate 和 FooterTemplate：包含在列表的开始和结束处呈现的文本和控件。
- SeparatorTemplate：包含在每项之间呈现的元素。典型的示例可能是一条直线(使用 HR 元素)。



设计者需要根据不同的需要定义不同类型的项模板，DataLis 控件根据项的运行时状态自动加载相应的模板显示数据。

(1) 启动 VWD 2010，新建网站【上机练习 5-2】。

(2) 向 Default.aspx 页面中添加一个 DropDownList 控件和一个 DataList 控件。

(3) 为 DropDownList 控件添加数据源，该数据源查询班级表 Class 中的记录，设置 DropDownList 控件的 DataTextField 属性为 Cname，DataValueField 属性为 Cno，并启用 AutoPostBack。

(4) 为 DataList 控件添加数据源，查询指定班级的学生信息，在配置数据源的 Select 语句时，添加 WHERE 子句，使 Cno 等于 DropDownList 控件的选定值。

(5) 通过【DataList 任务】面板编辑控件的显示模板，只编辑 ItemTemplate 即可，使得每个显示项的名称均为中文，然后编辑“性别”字段的 DataBindings，如图 5-22 所示，在打开的【SgenderLabel DataBindings】对话框中选中【自定义绑定】单选按钮，在【代码表达式】文本框中输入如下表达式，如图 5-23 所示。

```
Eval("Sgender").ToString() == "M" ? "男" : "女"
```

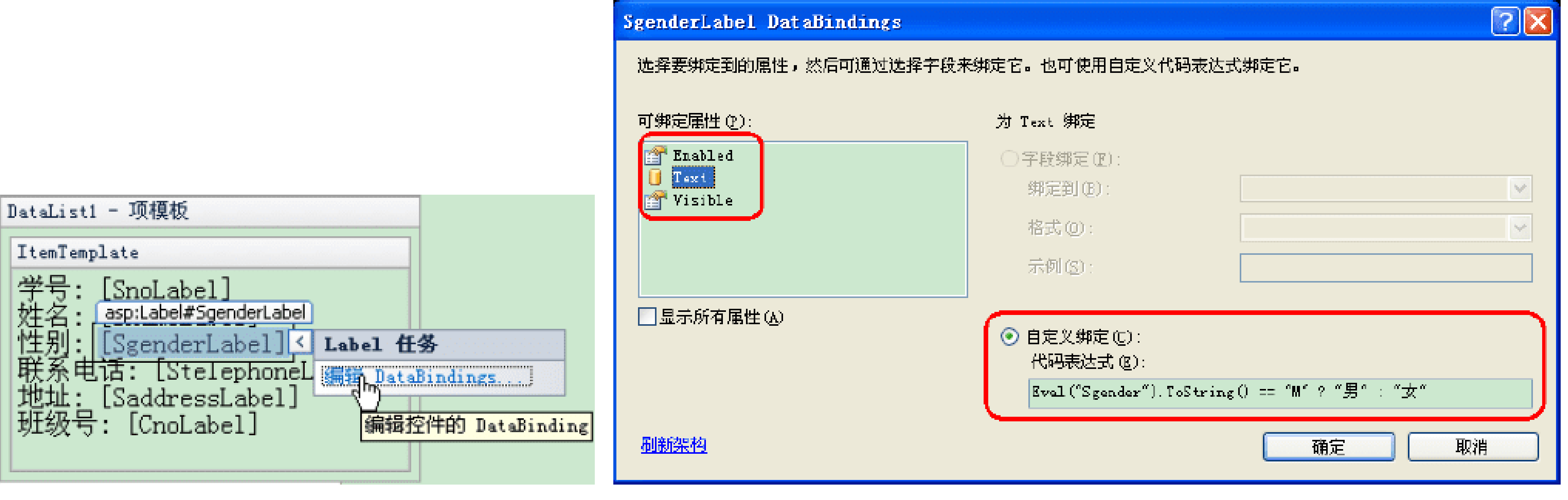


图 5-22　编辑 ItemTemplate 模板中的性别字段　　图 5-23　【SgenderLabel DataBindings】对话框

上述表达式的含义是，如果 Sgender 字段的值为“M”，则显示“男”，否则显示“女”。这是因为数据库中存放的“性别”字段是 char(1)类型，“M”表示男，“F”表示女。

(6) 编译并运行程序，通过下拉列表选择班级，DataList 控件将显示筛选后的该班级的学生信息，如图 5-24 所示。

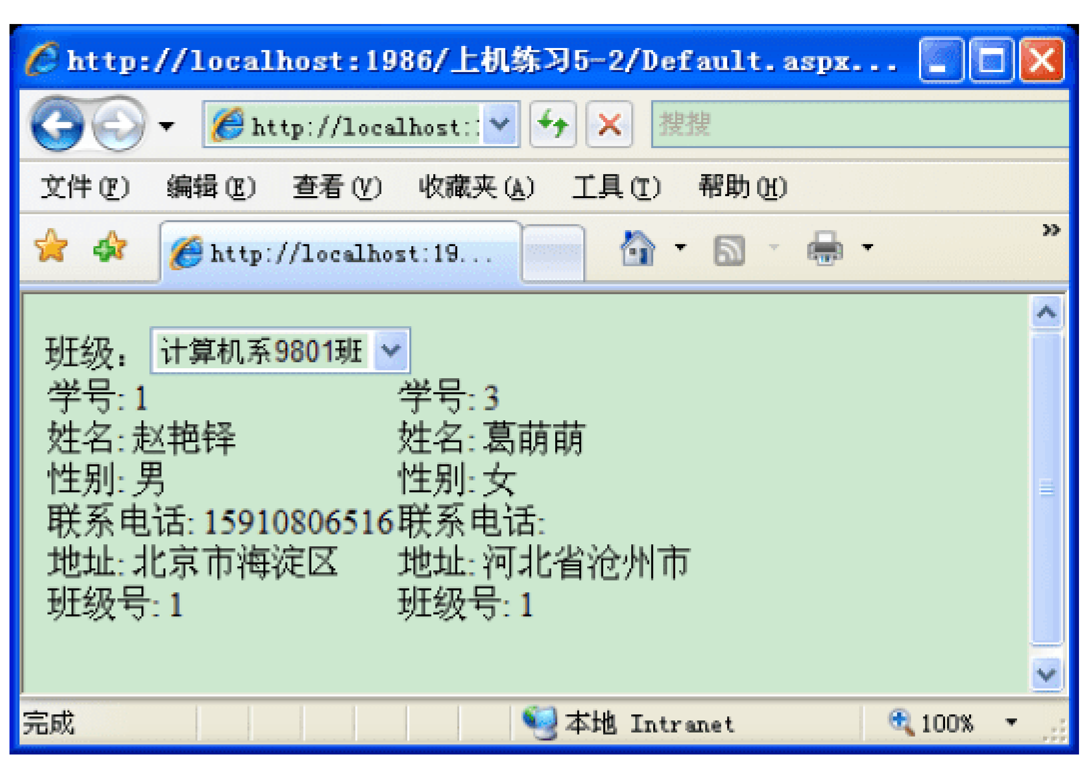
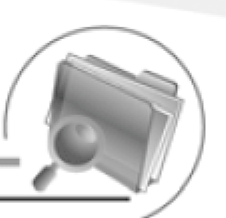


图 5-24　通过 DataList 控件筛选显示学生信息

5.5　习题

1. 编写程序，使用 ADO.NET 的提供者对象查询课程表中的所有课程信息。
2. 如何使用 ADO.NET 调用存储过程？
3. ASP.NET 提供了哪些数据源控件？
4. 编写程序，使用 GridView 控件显示指定课程的全部学生的成绩信息。
5. 要使用 GridView 控件的排序功能，需要设置什么属性？
6. 对于简单的数据绑定，必须调用页面或者控件的什么方法才能使绑定生效？
7. 如果想以下列格式显示数据库中歌曲流派的一个简单列表，需要选择哪种控件？

```
<ul>
  <li>Punk</li>
```



```
    <li>Hard Rock</li>
    <li>Jazz</li>
    <li>Techno</li>
</ul>
```

8. DetailsView 控件和 FormView 控件有什么相同点，又有什么区别？
9. DataKeyNames 属性有什么作用？





使用 LINQ

学习目标

LINQ 是一种与.NET Framework 中使用的编程语言紧密集成的新查询语言。使用 LINQ 可以直接通过代码查询多种数据源中的数据。本章主要介绍了 LINQ 语言及其语法，以及在 ASP.NET 项目中使用 LINQ 数据的许多方法。通过本章的学习，读者将掌握 LINQ 的基本语法以及 LINQ to EF 的具体应用。

本章重点

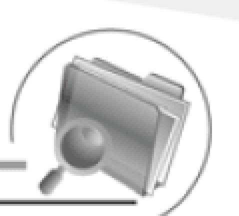
- LINQ 及其语法
- LINQ 的各种形式及其适用场合
- 了解 ADO.NET Entity Framework
- 使用 EntityDataSource 控件来访问 EF
- ListView 控件和 DataPage 控件的使用

6.1 LINQ 简介

LINQ，即语言集成查询(Language-Integrated Query)，是一种与.NET Framework 中使用的编程语言紧密集成的新查询语言。它使得用户可以像使用 SQL 查询数据库的数据那样从.NET 编程语言中查询数据。事实上，LINQ 语法部分模仿了 SQL 语言，它使得熟悉 SQL 的编程人员更容易上手。

使用 LINQ 可以直接通过代码查询多种数据源中的数据。LINQ 之于.NET 应用程序编程就像 SQL 之于关系数据库。通过简单的、声明性的语法，可以查询集合中匹配条件的对象。

LINQ 并不只是.NET Framework 的一个增件。相反，它被设计和实现为.NET 编程语言中的一部分。这意味着，LINQ 被真正集成到.NET 中，为查询数据提供了一个统一的方法，而不管数



据的来源。另外，由于它被集成到语言中，而不是特定的项目类型中，所以可用于各种项目，包括 Web 应用程序、Windows Forms 应用程序、Console 应用程序等。

LINQ 相关的类都放在 System.Linq 命名空间，所以要使用 LINQ，必须引入该命名空间：

```
using System.Linq;
```

由于 LINQ 非常强大，又极具潜力，因此它被集成到.NET Framewrok 的多个不同地方。下面将介绍不同的 LINQ 实现。



提示

LINQ 是在.NET 3.5 以后引入的，所以在.NET 2.0 及以前的版本中是不能使用 LINQ 的。

6.1.1　LINQ to Objects



这是语言集成的最基本形式。使用 LINQ to Objects，可以查询.NET Framework 中存在的几乎所有集合。实际上，使用 LINQ to Objects 对内存中的所以对象进行简单查询是非常方便的。

使用 LINQ 的查询通常由 3 个步骤组成：

(1) 获得数据源

(2) 创建查询

(3) 执行查询

【例 6-1】使用 LINQ 查询编号为奇数的用户信息，本例用到了泛型中的 Dictionary<K,V>定义一组集合。

(1) 启动 VWD 2010，新建网站【例 6-1】。

(2) 在 Default.aspx 页面的 Load 事件中添加如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Dictionary<int,string> user = new Dictionary<int,string>();
    user.Add(1, "赵艳铎");
    user.Add(2, "小石头");
    user.Add(3, "葛萌萌");
    user.Add(4, "金百合");
    var result = from val in user
                 where val.Key % 2 == 1
                 select val;
    foreach(var item in result)
    {
        Response.Write("编号："+ item.Key + "；        姓名："+item.Value);
        Response.Write("<br/>");
```



```
        }
    }
```

(3) 上述代码是查询编号为奇数的用户信息并输出。编译并运行程序，如图 6-1 所示。



图 6-1 页面运行效果



6.1.2 LINQ to XML

LINQ to XML 是读、写 XML 的一种新的.NET 方法。现在，可以在应用程序中编写直接针对 XML 的 LINQ 查询，而不是使用普通的 XML 查询语言，如 XSLT 或 XPath。

在 System.Xml.Linq 命名空间中定义了很多 LINQ to XML 的类，这些类的结构关系如图 6-2 所示。

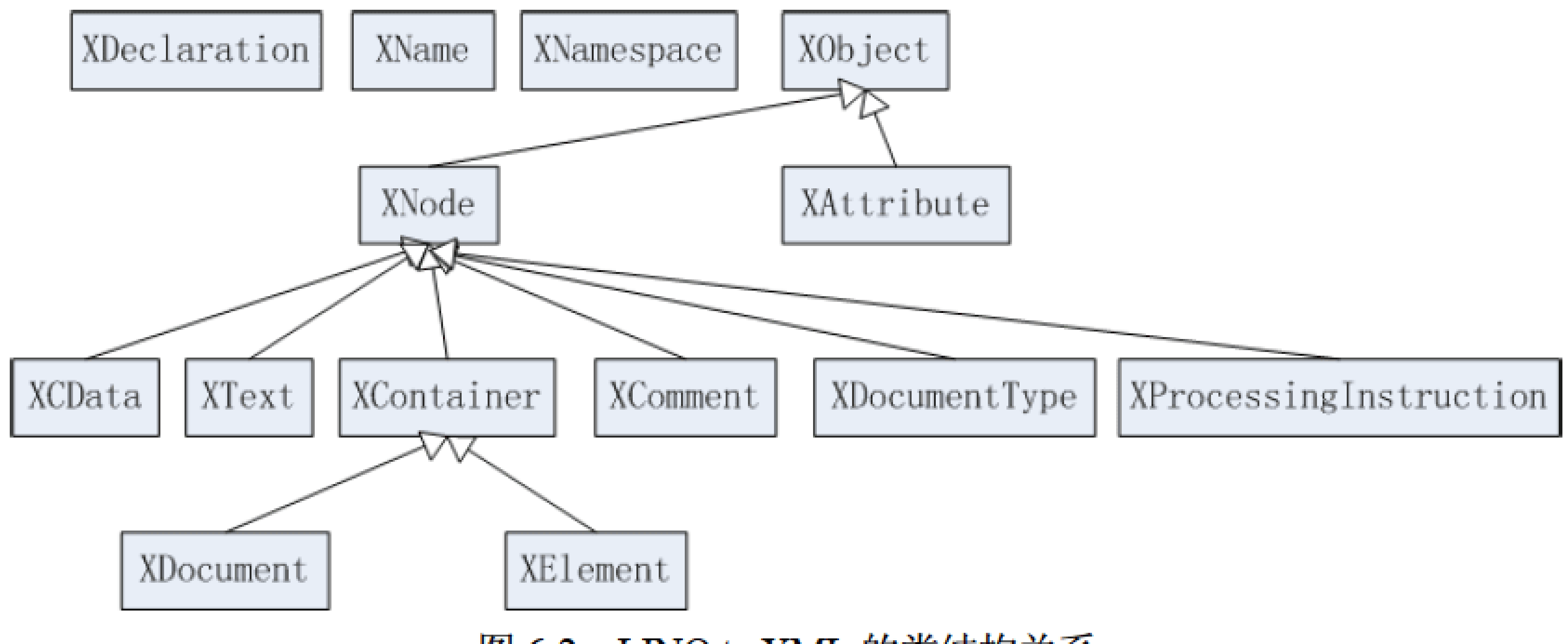


图 6-2 LINQ to XML 的类结构关系

其中 XElement 类是 LINQ to XML 中最基础的类，使用它可以创建一个 XMl 元素，使用 XAttribute 类可以为元素添加属性，使用 XNamespace 类可以为 XML 定义命名空间。

6.1.3 LINQ to ADO.NET

ADO.NET 是.NET Framework 的一部分，它允许访问数据、数据服务(像 SQL Server)和其他许多不同的数据源。使用 LINQ to ADO.NET，可以查询与数据库相关的信息集，包括 LINQ to Entities、LINQ to DataSet、LINQ to SQL 和 LINQ to Entities。



LINQ to Entities 是 LINQ to SQL 的超集，比后者有更丰富的功能。不过，对于大多不同类型的应用程序来说，LINQ to SQL 足够了。

LINQ to DataSet 允许对 DataSet 编写查询。

LINQ to SQL 允许在.NET 项目中编写针对 Microsoft SQL Server 数据库的面向对象的查询。LINQ to SQL 将 LINQ 查询转换为 SQL 语句，然后再发送到数据库中执行 CRUD 的 4 种操作。在 ASP.NET 4.0 中，Microsoft 已经表示不会再积极开发 LINQ to SQL，这是因为 LINQ to SQL 与 Entity Framework(EF)在功能上有很大的重叠，在 LINQ to SQL 中能实现的操作在 EF 中也能实现。但是，与 LINQ to SQL 相比，EF 的功能要强大得多，并且功能要丰富得多。正因为如此，Entity Framework 是比 LINQ to SQL 更好的选择。本章将重点讨论 Entity Framework。

6.1.4　LINQ 和泛型



LINQ 查询是建立在泛型这种数据类型的基础之上的，泛型是在.NET Framework 2.0 开始引入的。虽然编程人员无须深入了解泛型技术就可以开始编写 LINQ 查询，但是了解下面的两个泛型的基本概念有助于帮助读者理解其工作原理。

(1) 当创建泛型集合类(如 List<T>)的实例时，需要将“T”替换为集合中指定的对象类型。如字符串集合表示为 List<string>。因为泛型集合是强类型的，所以比将元素存储为 Object 类型的集合要强大得多。如果使图将一个 int 类型的对象添加到 List<string>，则会产生编译错误。

(2) IEnumerable<T>表示的是一个接口，通过该接口可以使用 foreach 语句来遍历泛型集合类。LINQ 查询变量可以类型化为 IEnumerable<T>或者它的派生类，如 IQueryable<T>。

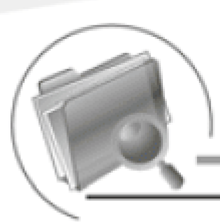
为了避免使用泛型语法，可以使用匿名类型来声明查询，即使用 var 关键字来声明查询。var 关键字指示编译器通过查看在 from 子句中指定的数据来推断查询变量的类型。如【例 6-1】中声明的 result 和 item 变量。

6.2　ADO.NET Entity Framework

通过使用 ADO.NET Entity Framework(EF)，可以把许多数据库对象(如表)转换成可以在代码中直接访问的.NET 对象，然后就可以在查询中或者直接在数据绑定中使用这些对象。EF 也允许执行相反的操作：首先设计一个对象模型，然后让 EF 创建必要的数据库结构。

使用 EF 十分简单，并且十分灵活。通过使用关系图设计器，可以将表等数据库对象拖放到实体模型中。放到关系图中的对象将成为可用的对象。例如，如果将 Student 表放到关系图中那么就将得到一个强类型的 Student 类。可以使用 LINQ 查询或者其他方式创建这个类的实例。

当把多个相关的数据库表放到关系图中时，设计人员可以观察到表之间的关系，然后在对象模型中复制这些关系。例如，如果使用某些 LINQ to EF 查询在代码中创建了一个 Student 实例，那么就可以访问它的 Cno 属性，进而可以访问 Cname 等属性：




```
Label1.Text = student.Cno.Cname;
```

类似地，也可以访问某个班级的所有学生集会 Students，以便将其绑定到数据绑定控件：

```
Repeater1.DataSource = Class.Students;
```

现在还不需要深究这些语法，本章的后面将详细介绍它们。下面通过一个具体的示例来介绍如何通过 EF 把数据模型映射到对象模型。

【例 6-2】创建 ADO.NET 实体数据模型，通过 LINQ 查询来访问底层表中的数据。

(1) 启动 VWD 2010，新建网站【例 6-2】。

(2) 在【解决方案资源管理器】面板中右击项目，从弹出的快捷菜单中选择【添加新项】命令，然后选择【ADO.NET 实体数据模型】模板，默认名称为 Model.edmx，单击【添加】按钮将其添加到项目中。此时会弹出如图 6-3 所示的提示框，提示该类文件通常放在 App_Code 文件夹中，单击【是】按钮将其放在 App_Code 文件夹中。也可以直接在 App_Code 文件夹上单击鼠标右击添加。



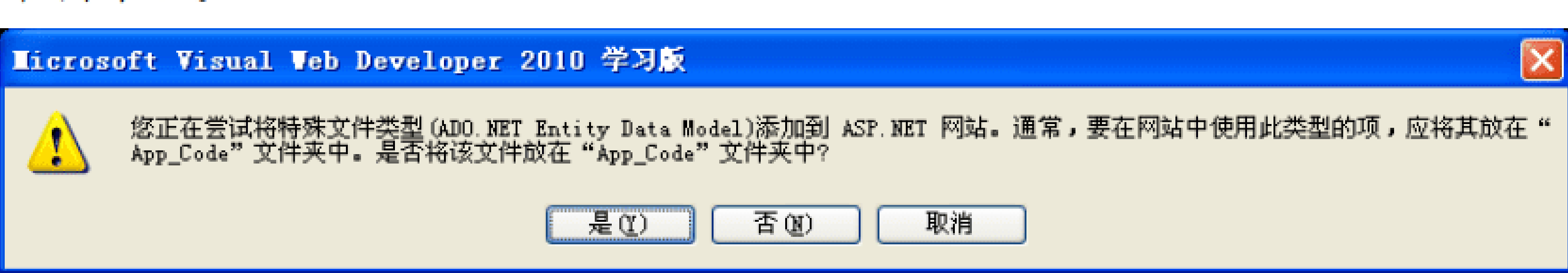


图 6-3　LINQ to XML 的类结构关系

(3) 系统将启动【实体数据模型向导】，第一步是【选择模型内容】，如图 6-4 所示。选择【从数据库生成】选项，然后单击【下一步】按钮。

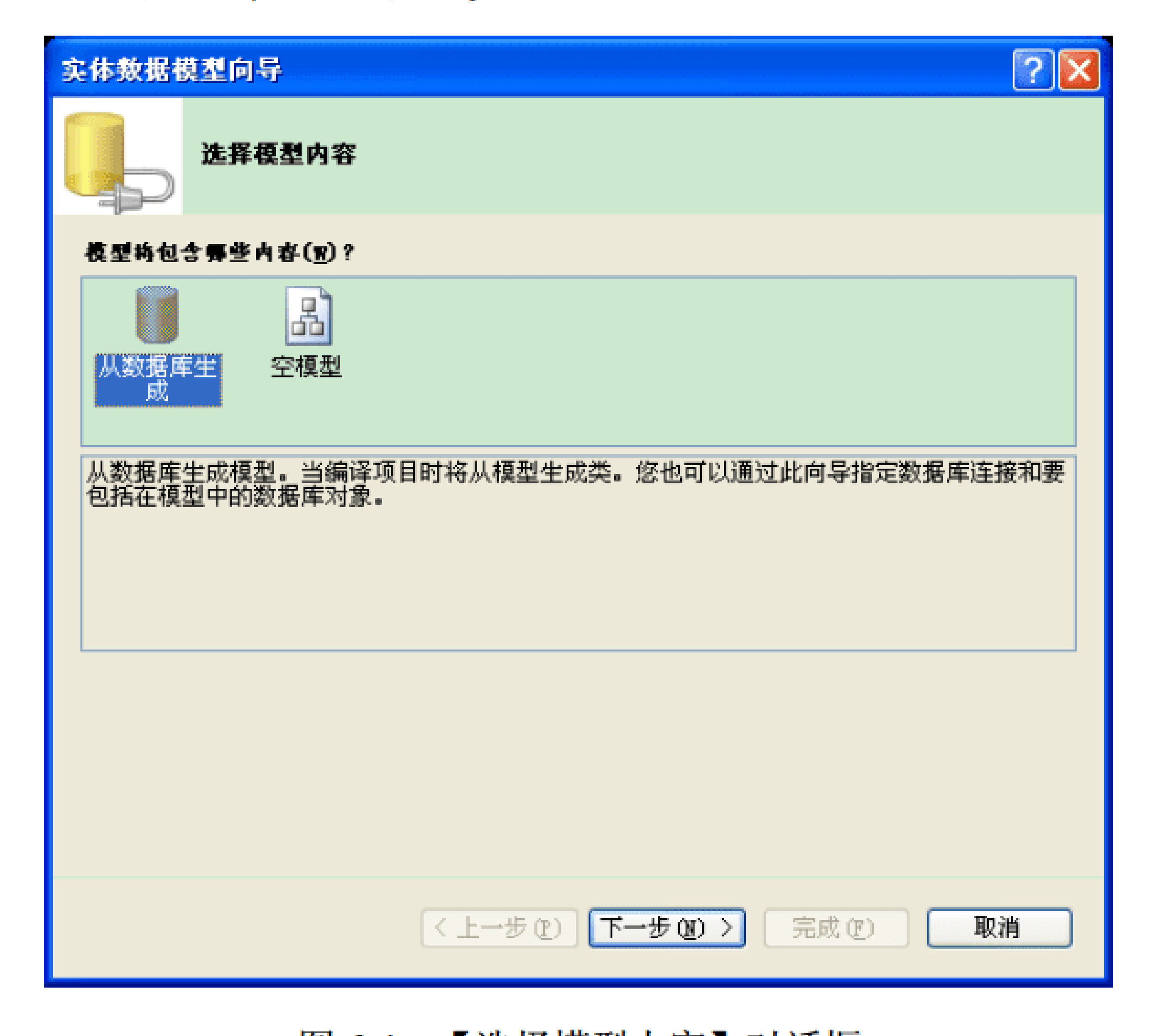


图 6-4　【选择模型内容】对话框



(4) 在【选择您的数据连接】对话框中，从下拉列表框中选择上一章中使用的 InfoManage 数据库连接，并选中【将 Web.Config 中的实体连接设置另存为】复选框，如图 6-5 所示。



图 6-5 【选择模型内容】对话框

(5) 单击【下一步】按钮进入【选择数据库对象】对话框，在该对话框中，选择表 Class 和 Student。该对话框下面有两个复选框，第一个表示在模型中自动将所有名称转换为单数或者复数形式，第二个是在模型中加入外键列，选中这两个复选框可以确保生成的对象模型中保留了表之间的关系，如图 6-6 所示。如果没有选中模型中自动将名称转换为单数或者复数的复选框，则需要在生成模型以后手动进行修改。另外，此处的模型命名空间需要记住，稍后编写代码时需要引用该命名空间。



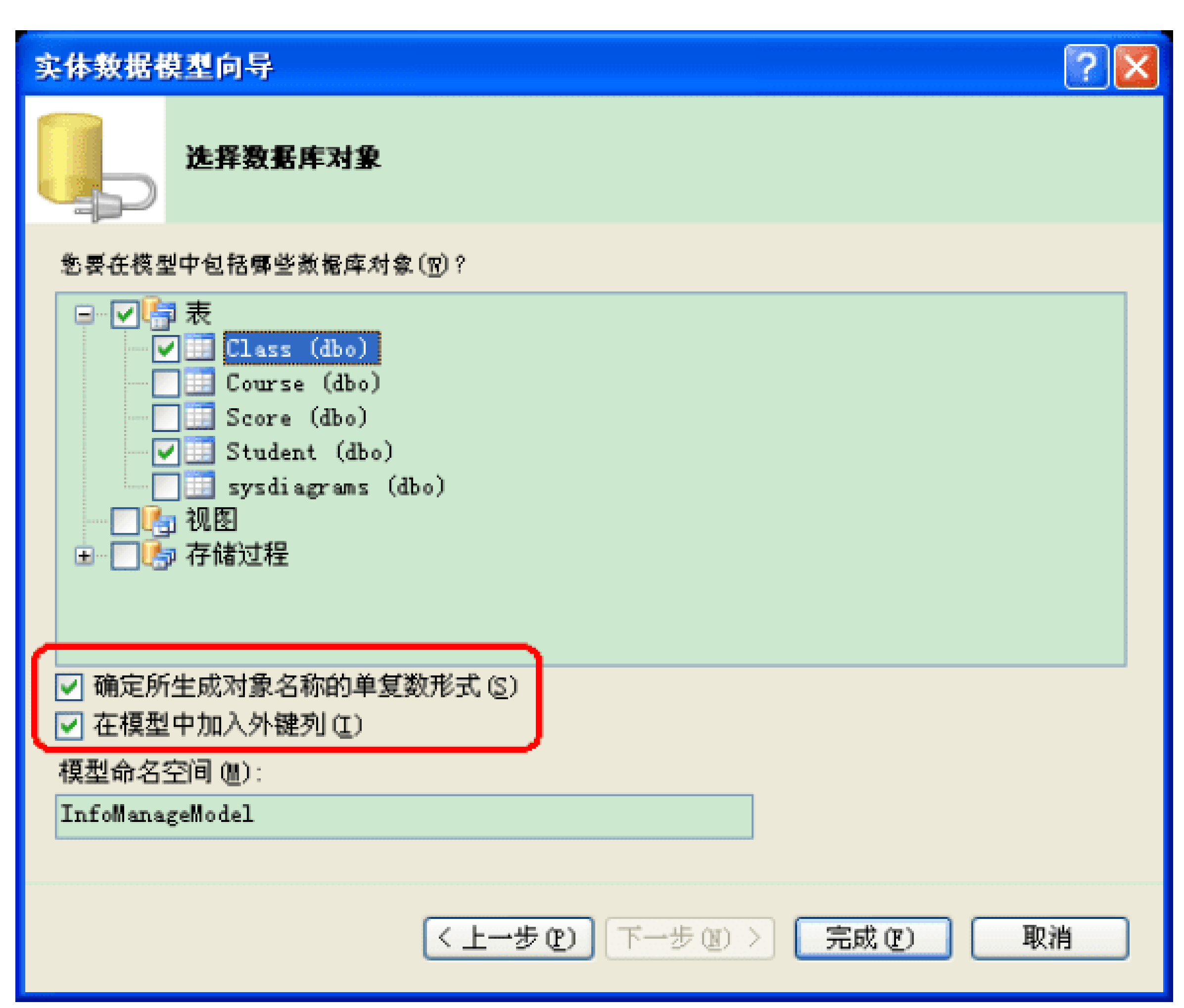


图 6-6 【选择数据库对象】对话框

(6) 单击【完成】按钮即可将模型添加到站点中。VWD 将添加一个 Model.edmx 文件和 Model.designer.cs 后台代码文件，然后在主编辑器窗口中打开【实体设计器】，如图 6-7 所示。

这个关系图显示了基于数据库表生成的类，两个类之间的线表示了底层表之间的关系。二者是一对多的关系，即一个班级中可以有多个学生，而每个学生只能归属一个班级。所以上图中 Class 类的导航属性是 Students(复数)，Student 的导航属性是 Class(单数)。



提示

如果前面没有选中自动将名称转换为单数或者复数的复选框，那么在此可以分别选中生成的类，然后在【属性】面板中将【实体集名称】改为复数形式；然后，在类图中将导航属性重命名为复数形式。导航属性的修改需要根据具体的关系，对于一对多的关系中的“1”方不用修改为复数，而【实体集名称】则是所有类都要修改。



(7) 在 Default.aspx 页面中添加一个 DropDownList 控件和一个 GridView 控件，设置 DropDownList 控件的 AutoPostBack 属性为 True。

(8) 在页面的 Load 事件处理程序中设置 DropDownList 控件的数据源为班级信息，添加如下代码:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        using (InfoManageEntities myEntity = new InfoManageEntities())
        {
            var allClass = from c in myEntity.Classes
                           select c;
            DropDownList1.DataSource = allClass;
            DropDownList1.DataTextField = "Cname";
            DropDownList1.DataValueField = "Cno";
            DropDownList1.DataBind();
        }
    }
}
```

提示

当输入 InfoManageEntities 时会显示一个错误，因为它是定义在作用域之外的一个命名空间中的。这时需要在文件顶部引入图 6-6 中提到的命名空间 using InfoManageModel;。

(9) 添加 DropDownList 控件的 SelectedIndexChanged 事件处理程序，代码如下:



```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    int cno = Int32.Parse(DropDownList1.SelectedValue);
    using (InfoManageEntities myEntity = new InfoManageEntities())
    {
        var stu = from student in myEntity.Students
                  where student.Cno == cno
                  select student;
        GridView1.DataSource = stu;
        GridView1.DataBind();
    }
}
```

(10) 编译并运行程序，在默认浏览器中加载 Default.aspx 页面，在下拉列表中选择某个班级，下方的 GridView 控件中将显示相应班级的学生信息，如图 6-8 所示。

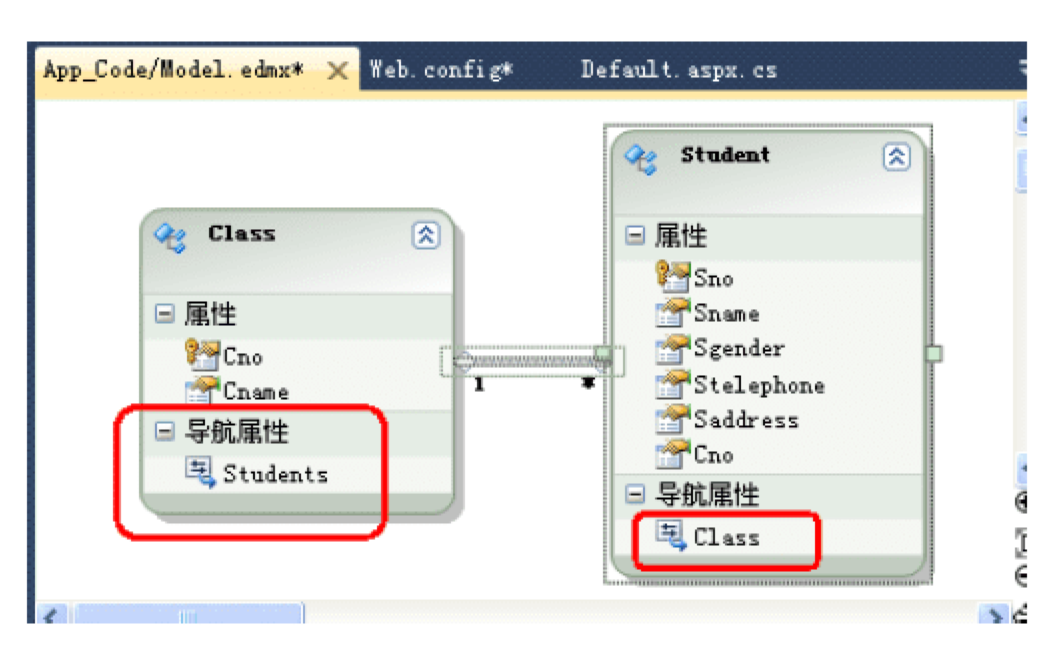


图 6-7　生成的实体对象模型关系图

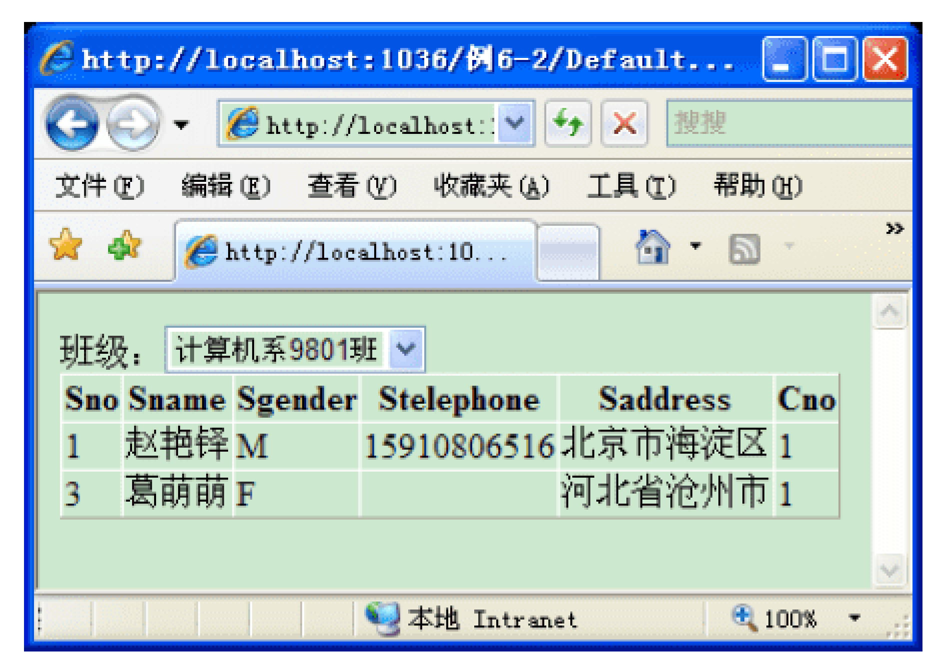


图 6-8　显示指定班级的学生信息

通过这个例子可以看出，EF 提供了一个对象关系设计器(可以通过 VWD 访问)，允许基于数据库的表创建一个可通过代码访问的对象模型。只要将表拖至该设计器，VWD 就会创建可用于访问数据库中底层数据的代码，而无需自己编写大量代码。拖至设计器上的类存储在.edmx 文件和其后台代码文件中。设计器文包含了一个继承自 ObjectContext 的类，而 ObjectContext 是 EF 中提供对数据库进行访问的主要实体。

在生成模型后，对其执行 LINQ 查询，从而从底层数据库中获取数据。要访问这些数据，需要一个 ObjectContext 类的实例，这在代码的 using 块中创建。using 中包装的代码用于创建在用完后必须释放(从内存中清除)的变量。由于 myEntity 保存了到 SQL Server 数据库的连接，因此将使用它的代码包装到 using 块中，这样对象会在块的末尾销毁。

6.3　LINQ 查询语法

前面的例子中使用了简单的 LINQ 查询，LINQ 的查询能力远高于此。本节将重点介绍 LINQ

查询语法。需要注意的是：LINQ 语法并不是专门为 Entity Framework 设计的。下面介绍的大多数 LINQ 概念同样适用于其他 LINQ 实现，如 LINQ to Objects 和 LINQ to ADO.NET 等。

6.3.1 基本语法

LINQ 支持大量的查询操作符——可用于选择、排序或筛选从查询返回的数据的关键字。尽管本章所有示例是在 LINQ to EF 的背景下讨论的，但也可以将它们应用到其他 LINQ 实现中。接下来将通过示例介绍一些最为重要的标准查询操作符。每个示例都使用对象模型和【例 6-2】中创建的名为 myEntity 的 ObjectContext 对象作为查询的数据源。

1. from

LINQ 查询表达式必须以 from 子句开头。尽管 from 子句不能算是标准查询操作符，因为它并不对数据进行操作而是指向数据，但它是 LINQ 查询中的一个重要元素，因为它定义了查询所执行的集合或数据源。



2. select

select 关键字用于从查询的源中检索对象。在这个示例中，可看到如何选择已有类型的一个对象。

```
var allClass = from c in myEntity.Classes
               select c;
```

这一示例中的变量 c 指范围变量(range variable)，它只在当前查询中可用。通常在 from 子句中引入范围变量，然后在 where 和 select 子句再次使用它来筛选数据，表明要选择的数据。尽管对于它可采用任意的名称，但通常看到的都是单个字母的变量，如 c，或所查询集合的单数形式 student。

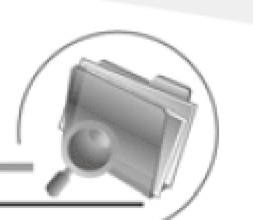
3. where

和 SQL 中的 WHERE 子句一样，LINQ 中的 where 子句允许筛选查询返回的对象。下列的查询将返回指定班级的学生：

```
var stu = from student in myEntity.Students
          where student.Cno == cno
          select student;
```

4. orderby

使用 orderby 可以对结果集合中的项进行排序。orderby 后面可以通过逗号分隔来指定多个条件。紧跟着的是可选的用来指定排序顺序的 ascending(升序)和 descending(降序)关键字，默认排序方式为升序。例如下列的查询将把结果按 Sname 列降序排列：



```
var stu = from student in myEntity.Students
          where student.Cno == cno
          orderby student.Sname descending
          select student;
```

5. Sum、Min、Max、Average 和 Count

这些聚集运算符允许在结果集中的对象上进行数学计算。Sum 是求和运算符；Min 是求最小值运算符；Max 是求最大值运算符；Average 是求平均值运算符；Count 是计数运算符。例如，要检索指定班级的学生人数，可以执行如下查询：

```
var stu = (from student in myEntity.Students
          where student.Cno == cno
          orderby student.Sname descending
          select student).Count();
```



6. Take、Skip、TakeWhile 和 SkipWhile

Take 和 Skip 允许在结果集中作子选择。这很适用于分页情况，其中只检索当前页面的记录。Take 从结果集中获取所请求数量的元素，然后忽略其余的；而 Skip 则相反，它跳过请求数量的元素，然后返回其余的。

在 EF 中，Take 和 Skip 操作符也被转换为 SQL 语句。这意味着分页是在数据库级发生的，而不是在 ASP.NET 页面中。这大大增强了查询的性能，特别是对于一些较大的结果集更是如此，因为不是所有的元素都必须从数据库转移到 ASP.NET 页面中。



知识点

要想使用 Skip，必须在跳过指定数量的记录之前，向查询中添加一个 orderby 子句来对结果进行排序。如果不显式地添加 orderby 子句，数据库就可能以无法预知的顺序返回结果，因此，在 LINQ 查询中添加 orderby 动作有助于从 Skip 方法获得一致的结果，因为在跳过和获取记录之前，会先对它们进行排序。

下面的例子显示了如何检索第二页的记录，假定页面大小为 10。

```
var stu = (from student in myEntity.Students
          orderby student.Sname descending
          select student).Skip(10).Take(10);
```

和 Count 一样，该查询也被括在一对括号中，然后调用 Skip 和 Take 来获取请求的记录。

TakeWhile 和 SkipWhile 查询操作符的工作方式类似，但允许在特定条件满足时获取或跳过一些记录。遗憾的是，在 EF 中无法使用它们，但是通常可以通过给查询添加一个简单的 Where 子句来解决这个问题。

7. Single 和 SingleOrDefault

Single 和 SingleOrDefault 操作符允许返回单个对象作为强类型化实例。如果已知查询只返回一条记录，将很有用；例如，通过 Sno 查询学生信息：

```
var stu = (from student in myEntity.Students
           where student.Sno == 1
           select student).Single();
```

如果请求的项未找到或是查询返回多个实例，Single 操作符就会引发异常。如果想让该方法返回 null(没找到)，或是返回相关数据类型的默认值(如 Integer 型的 0、Boolean 型的 False 等)，则使用 SingleOrDefault。



提示

即使查询结果只有一条记录，如果未调用 Single，则仍会返回一个列表集合。通过使用 Single，可强制结果集为所查询类型的单个实例。

8. First、FirstOrDefault、Last 和 LastOrDefault

这些操作符允许返回特定对象序列对象中的第一个或最后一个元素。和 Single 方法一样，如果集合为空，First 和 Last 就会抛出异常，而 FirstOrDefault 和 LastOrDefault 则返回相关数据类型的默认值。

与 Single 不同的是，当查询返回多个项时，First、FirstOrDefault、Last 和 LastOrDefault 操作符并不抛出异常。

但是， EF 中并不支持 Last 和 LastOrDefault 查询。不过，通过使用 First 和降序排列可以实现与之相同的操作。

6.3.2 用匿名类型定型数据

到目前为止，在前面几节中看到的查询都返回的是全类型。即，查询返回了一个 Student 实例的列表(例如 select 方法)或是一个数值(如 Count)。

不过，有的时候我们不需要这些对象的所有信息，或者想对某些信息进行映射或转换。例如在例 6-2 中，希望在 GridView 的标题列显示中文描述信息，在“性别”列显示“男”或“女”。这时，可以通过匿名类型(anonymous type)来重新定义要显示的列。

匿名类型是一种不需要像使用其他类型(如类)时那样预先定义名称的类型。而是可以通过选择数据，然后让编译器推断其类型来进行构造。

创建匿名类型很简单，不需要使用像 select student 这样的语句选择实际的对象，而是使用 new 关键字，然后在一对花括号中定义要选择的属性。例如：



```
            var stu = from student in myEntity.Students
                      where student.Cno == cno
                      select new{studnet.Sno,student.Sname};
```

尽管这一类型是匿名的，不能通过名称直接访问，但编译器仍能推断其类型，对于在查询中选择的新属性提供了完全智能识别感知功能。

除了直接选择已有属性外，还可以创建属性值并提供不同的名称。例如，修改【例 6-2】中查询学生信息的 LINQ 语句，创建一个新的匿名类型，重命名所有列，在“性别”列显示“男”或“女”，并在“班级”列显示班级名称。修改后的代码如下：

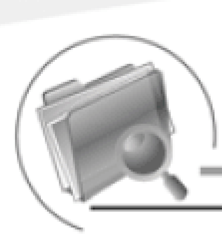
```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    int cno = Int32.Parse(DropDownList1.SelectedValue);
    using (InfoManageEntities myEntity = new InfoManageEntities())
    {
        var stu = from student in myEntity.Students
                  where student.Cno == cno
                  select new
                  {
                      学号 = student.Sno,
                      姓名 = student.Sname,
                      性别 = (student.Sgender=="M"?"男":"女"),
                      联系电话 = student.Stelephone,
                      地址 = student.Saddress,
                      班级 = student.Class.Cname
                  };
        GridView1.DataSource = stu;
        GridView1.DataBind();
    }
}
```

页面的运行效果如图 6-9 所示。



图 6-9　使用匿名类定型数据后的效果







6.4 使用数据控件和 LINQ

在前面的例子中，将 LINQ 查询的结果指派给控件的 DataSource 属性，然后调用 DataBind 方法即可显示在页面中。但是，这种方法只能显示数据，它不支持直接编辑、更新和删除数据。本节将介绍 EntityDataSource 控件和数据控件的绑定，使用这些控件可以很容易地实现编辑、更新和删除功能。

6.4.1 EntityDataSource 控件

EntityDataSource 和 SqlDataSource 及其他数据源控件类似。EntityDataSource 控件之于 EF 就像 SqlDataSource 控件之于基于 SQL 的数据源：它提供了一个声明性的方法来访问支持 LINQ 的数据源模型。和 SqlDataSource 控件一样， EntityDataSource 提供了对 CRUD 操作的轻松访问，另外使数据排序和筛选也变得非常简单。EntityDataSource 控件的主要属性如表 6-1 所示。



表 6-1 EntityDataSource 控件的主要属性

属　性	描　述
EnableDelete EnableInsert EnableUpdate	表明确定控件是否提供自动插入、更新和删除功能。如果启用，可以结合使用该控件和数据绑定控件(如 GridView 或 ListView)来支持数据管理。后面会介绍其应用
ContextTypeName	控件将使用的 ObjectContext 类的名称。在本书的示例中，这个类型名为 PlanetWrox DataContextEntities
EntitySetName	想使用的 EF 关系图中的表实体集名，如 Reviews
Select OrderBy Where	允许定义 EntityDataSource 控件对模型触发的查询。每个属性都映射到前面见到过的一个查询操作

和数据绑定控件一起， EntityDataSource 通过 EF 提供了对底层 SQL Server 数据库的完全访问。【例 6-3】将显示了如何在 ASPX 页面中使用该控件。

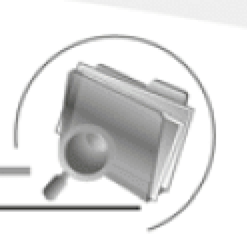
【例 6-3】使用 EntityDataSource 作为数据控件的数据源。

(1) 启动 VWD 2010，新建网站【例 6-3】。

(2) 按例 6-2 介绍的方法添加【ADO.NET 实体数据模型】Model.edmx，在【选择数据库对象】一步中，选择 Score 和 Student 表。

(3) 在 Default.aspx 页面的设计视图中添加一个 EntityDataSource 控件，在【EntityDataSource 任务】面板中单击【配置数据源】链接，开始配置数据源。

(4) 第一步是【配置 ObjectContext】，在此选择刚创建的实体数据模型的命名连接 InfoManageEntities，如图 6-10 所示。



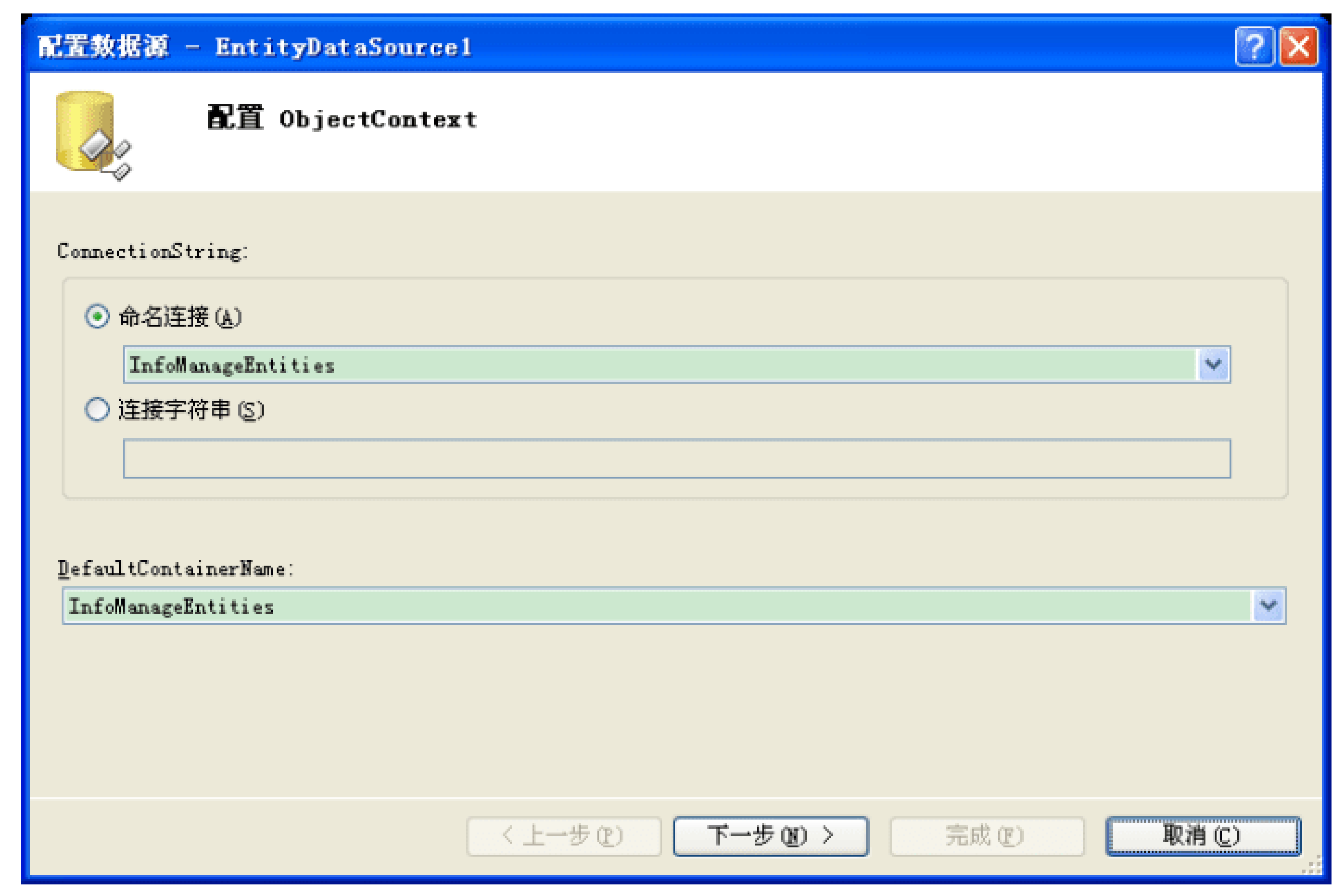


图 6-10　【配置 ObjectContext】对话框

(5) 单击【下一步】按钮，在【配置数据选择】对话框中选择 Students 实体集，并选中下面的 3 个复选框，如图 6-11 所示。

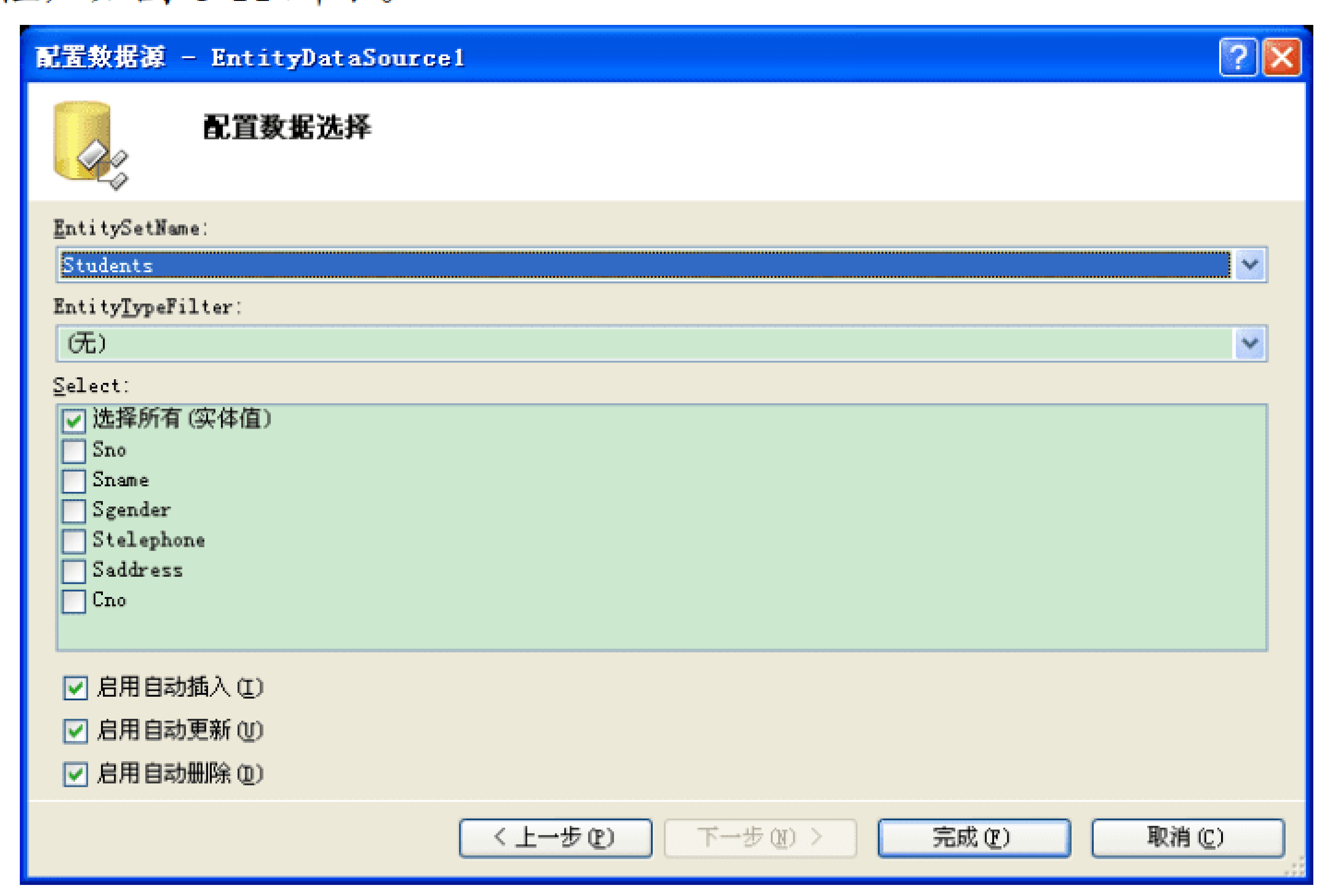


图 6-11　【配置数据选择】对话框



(6) 单击【完成】按钮，完成数据源的配置。

(7) 在 Default.aspx 页面中添加一个 DetailsView 控件。设置控件的数据源为前面创建的数据源 EntityDataSource1，并选中【DetailsView 任务】面板中的【启用分页】、【启用插入】、【启用编辑】和【启用删除】复选框，然后通过【编辑字段】选项，编辑各列的 HeaderText 属性为中文描述。

(8) 无须编写任何代码，编译并运行程序，在浏览器中加载 Default.aspx 页面，效果如图 6-12 所示。

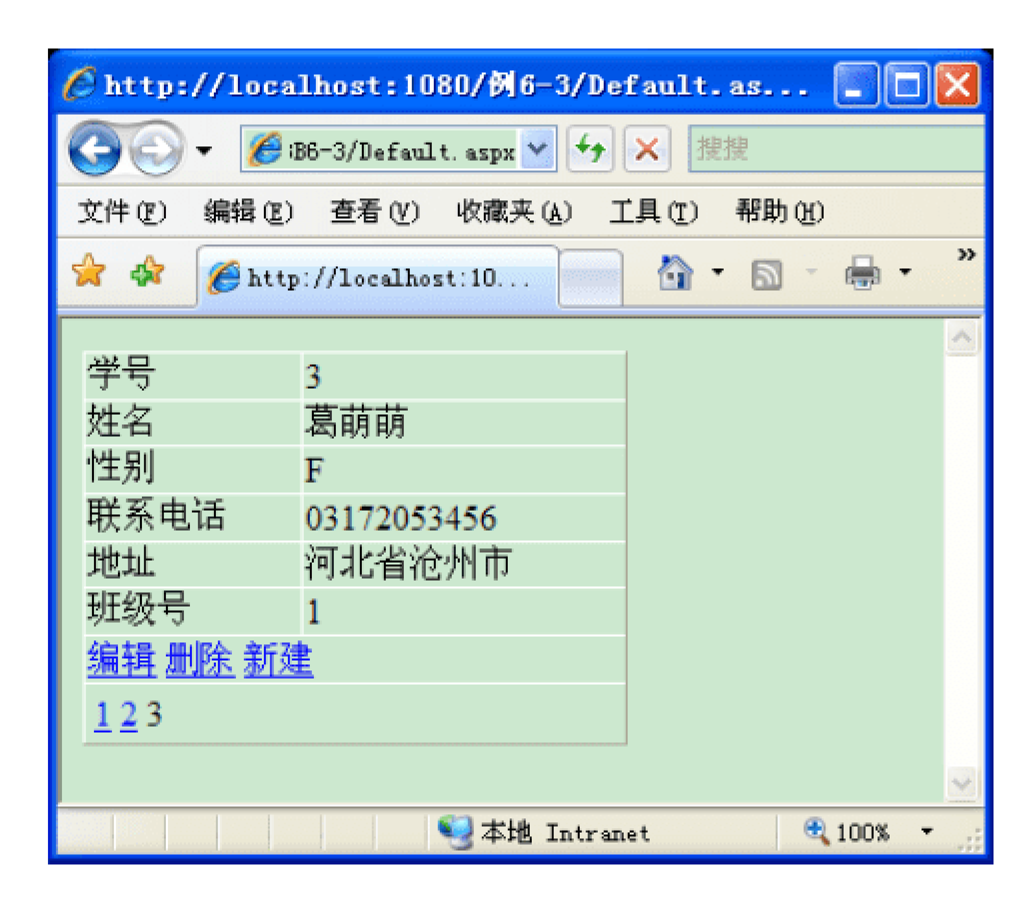


图 6-12 页面运行效果



通过这个示例可以看出，创建和使用 EntityDataSource 数据源控件和事业 SqlDataSource 控件一样简单。下一节将介绍使用 ListView 和 DataPager 控件显示指定学生的成绩信息。

6.4.2 使用 ListView 和 DataPager 控件

ListView 控件结合了 GridView 丰富的功能集和对 Repeater 提供的对标记的控制功能，并且还具有 DetailsView 的插入行为。ListView 使得可以以不同的格式显示数据，包括网格(像 GridView 那样的行和列)、项目符号列表、流格式等。

ListView 通过模板来显示和管理其数据。所有可用的模板如表 6-2 所示。

表 6-2 ListView 控件的可用模板

模　板	描　述
<LayoutTemplate>	作为控件的容器。它使得可定义一个放置单独数据项的位置。然后通过 ItemTemplate 和 AlternatingItemTemplate 表示的数据项作为该容器的子元素添加
<ItemTemplate> <AlternatingItemTemplate>	定义控件的只读模式。当一起使用时，它们可以创建一种“斑马纹效果”，其中奇偶行有着不同的外观(通常是不同的背景色)
<SelectedItemTemplate>	允许定义当前活动的或选择的项的外观
<InsertItemTemplate> <EditItemTemplate>	这两个模板允许定义用于插入和更新列表中的项的用户界面。通常，放置文本框、下拉列表和其他服务器控件等到这些模板中，将它们与底层数据源绑定
<ItemSeparatorTemplate>	定义放置在列表中项之间的标记。可用于在项之间添加线、图像或其他标记
<EmptyDataTemplate>	在控件中没有数据显示时显示。可以添加文本或其他标记，告诉用户无数据显示
<GroupTemplate> <GroupSeparatorTemplate> <EmptyItemTemplate>	在高级表现场景中使用，其中数据可呈现在不同的组中



尽管这些模板看上去让人觉得需要编写大量代码来使用 ListView，但事实并非如此。首先，VWD 2010 根据一些控件(如 EntityDataSource)提供的数据，创建了大部分的代码；其次，并不总是需要所有的模板，这就可以最小化控件所需的代码。

除了模板之外，ListView 控件还有很多属性，可以通过对其进行设置来影响控件的行为，如表 6-3 所示列出了 ListView 控件的主要属性。

表 6-3　ListView 控件的主要属性

模　板	描　述
ItemPlaceholderID	放置在 LayoutTemplate 中的服务器端控件的 ID。当该属性引用的控件在屏幕上显示时，将由所有重复的数据项取代。它可以是一个真正的服务器控件，如<asp:PlaceHolder>；或者是一个简单的 HTML 元素，带有一个有效的 ID，其 runat 特性设置为 server(例如<ul runat="server" id=" MainList" >)。如果不设置该属性，ASP.NET 会尝试找到 ID 为 itemPlaceholder 的控件并使用该控件
DataSourceID	页面上数据源控件的 ID，如 EntityDataSource 或 SqlDataSource 控件
InsertItemPosition	这一属性的枚举包括 3 个值(None、FirstItem 和 LastItem)，允许用于确定 InsertItem Template 的位置：在列表的开始或末尾，或者不可见



和其他数据绑定控件一样，ListView 也有大量的在控件生命周期的特定时间触发的事件。例如，它有在项插入到底层数据源前后触发的 ItemInserting 和 ItemInserted 事件。类似地，也有在更新和删除数据前后触发的事件。

ListView 控件的分页功能是通过 DataPager 控件实现的。DataPager 控件是 ASP.NET 3.5 引入的控件，可用它来扩展另一个数据绑定的控件。目前，只允许使用 DataPager 为 ListView 控件提供分页功能。

有两种方法将 DataPager 与 ListView 控件关联：可以在 ListView 控件的<LayoutTemplate>中定义它，或是完全在 ListView 的外部定义它。在第一种情况下，DataPager 知道应自动为给哪个控件提供分页功能。在第二种情况下，需要将 DataPager 的 PagedControlID 属性设置为有效 ListView 控件的 ID。下面将介绍如何结合 ListView 配置和使用 DataPager。

【例 6-4】在【例 6-3】的基础上，使用 ListView 控件和 DataPager 控件分页显示当前学生的成绩信息。

(1) 启动 VWD 2010，打开网站【例 6-3】。

(2) 向 Default.aspx 页面中添加一个 ListView 控件，然后为其配置数据源，选择【<新建数据源>】选项，启动数据源配置向导。

(3) 在【选择数据源类型】对话框中选择【Entity】类型，与配置 EntityDataSource1 类似，所不同的是这次选择的实体集是 Scores。

(4) 配置完数据源以后，选中刚创建的数据源 EntityDataSource2，在【属性】面板中单击“Where”属性后面的【...】按钮，打开【表达式编辑器】对话框，在该对话框中选中【基于

提供的参数自动生成 Where 表达式】复选框，然后单击【添加参数】按钮，添加参数 Sno，设置【参数源】为 DetailsView 控件，如图 6-13 所示，生成的数据源控件的代码如下所示。

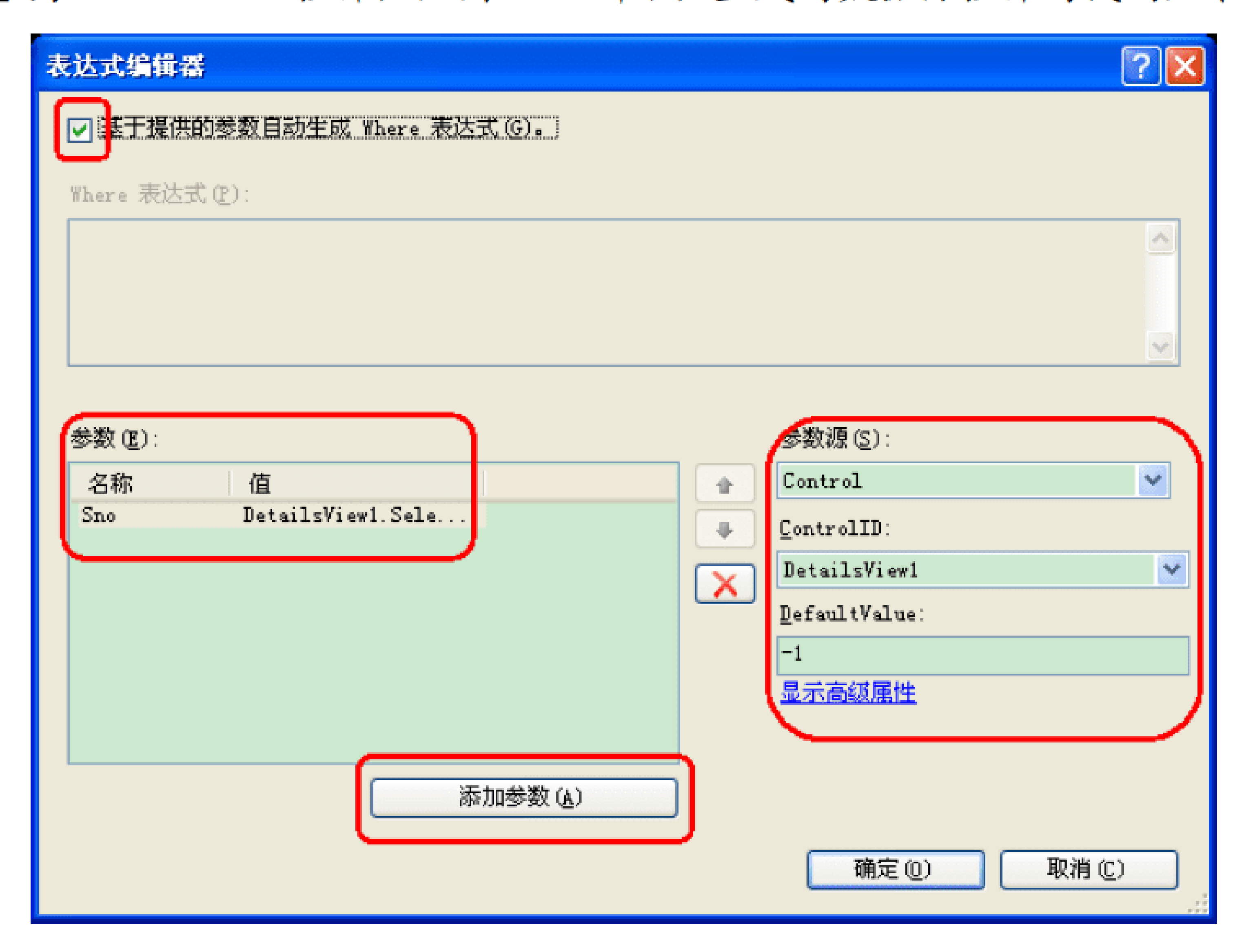
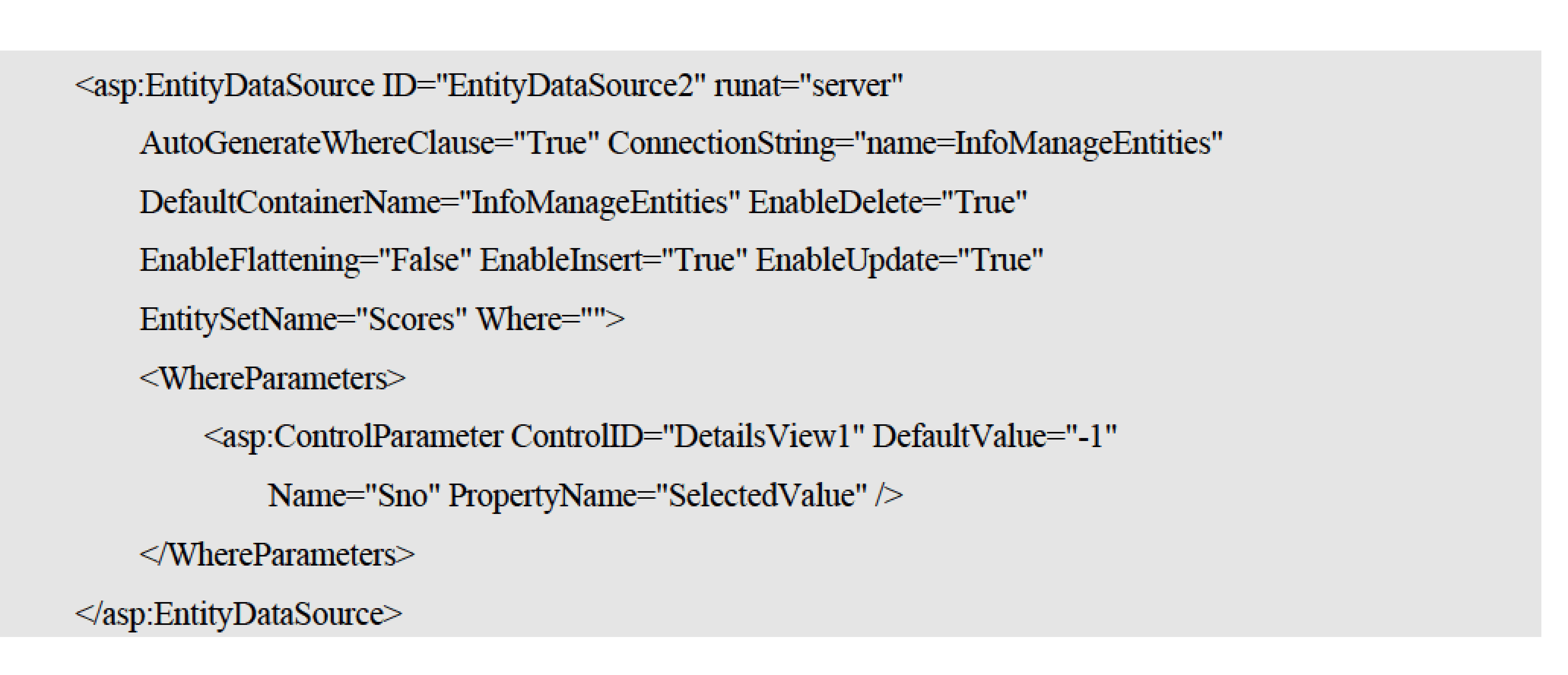


图 6-13 【表达式编辑器】对话框



```
<asp:EntityDataSource ID="EntityDataSource2" runat="server"
    AutoGenerateWhereClause="True" ConnectionString="name=InfoManageEntities"
    DefaultContainerName="InfoManageEntities" EnableDelete="True"
    EnableFlattening="False" EnableInsert="True" EnableUpdate="True"
    EntitySetName="Scores" Where="">
    <WhereParameters>
        <asp:ControlParameter ControlID="DetailsView1" DefaultValue="-1"
            Name="Sno" PropertyName="SelectedValue" />
    </WhereParameters>
</asp:EntityDataSource>
```

(5) 在【ListView 任务】面板上选择【配置 ListView】选项。出现的对话框允许选择控件的布局、样式和是否启用插入、更新和分页等操作。布局选择【网格】，样式选择【专业型】，同时选中【启用编辑】、【启用插入】、【启用删除】和【启用分页】复选框。单击【确定】按钮关闭该对话框。如果有对话框询问是否想重新生成 ListView 控件，则单击【是】按钮。

(6) 切换至源视图，删除所有模板中的<Student>列，修改 LayoutTemplate 模板中的列标题为中文描述。

(7) 定位到 LayoutTemplate 标记，找到其中的 DataPager 控件的定义，添加一个 PageSize 特性并设置其值为 3。代码如下所示:

```
<asp:DataPager ID="DataPager1" runat="server" PageSize="3">
    <Fields>
        <asp:NextPreviousPagerField ButtonType="Button" ShowFirstPageButton="True"
            ShowLastPageButton="True" />
    </Fields>
</asp:DataPager>
```



知识点

DataPager 控件的 PageSize 属性默认值为 10，因为数据库中记录少于 10 条，为了演示分页功能此处将其设置其为 3。

(8) 编译并运行程序，页面运行效果如图 6-14 所示。

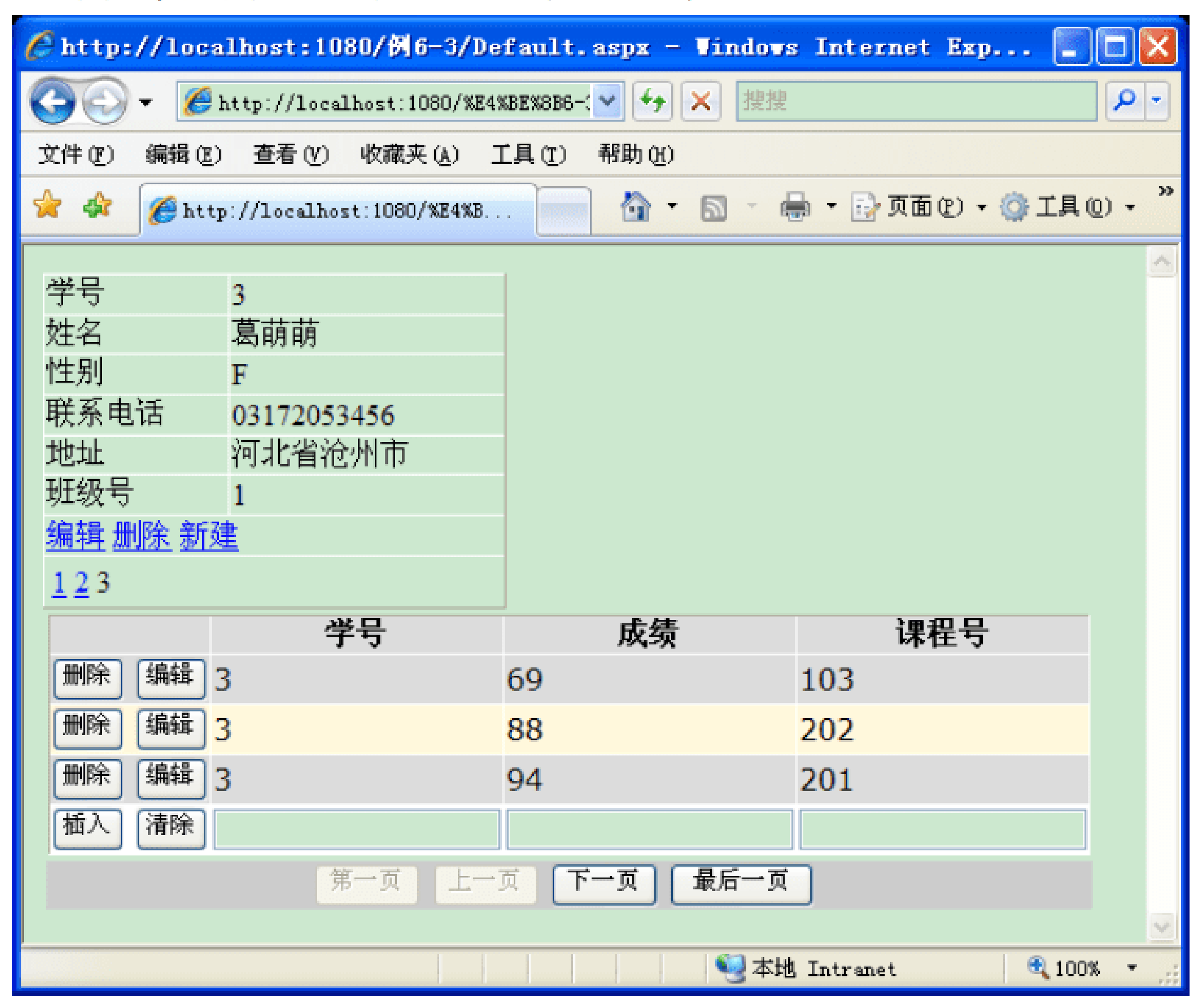
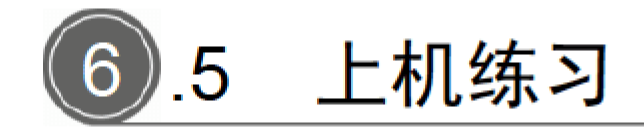


图 6-14　页面运行效果

本例中 DetailsView 控件支持插入、编辑、删除和分页功能，通过页码选择不同的学生，下面的 ListView 控件将显示该学生的成绩信息。ListView 控件也支持插入、编辑、删除和分页功能。通过该页面，可以维护学生信息和成绩信息。



6.5　上机练习

在前面的例子中，插入新记录时，对于“性别”列是要求用户输入文本，而数据库中存放的是“F”(女)和“M”(男)， 显然，让用户直接输不是非常友好，而且容易出错，理想的情况是



让用户选中单选按钮。下面的上机练习将介绍如何实现这种功能。

VWD 根据 EntityDataSource 中的信息生成的 ListView 控件的默认模板只适合于最常见的几种情况。通常，用户需要更多的控制。本章的上机练习就是通过自定义 ListView 控件的 InsertItemTemplate，让用户选择“性别”。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 6】。

(2) 按前面介绍的方法添加【ADO.NET 实体数据模型】Model.edmx，在【选择数据库对象】一步中，选择 Class 和 Student 表。

(3) 向 Default.aspx 页面中添加一个 ListView 控件，然后为其配置数据源，选择【<新建数据源>】选项，启动数据源配置向导。

(4) 在【选择数据源类型】对话框中选择【Entity】类型，选择实体集 Student，配置完数据源以后，系统将创建数据源 EntityDataSource1。

(5) 返回 Default.aspx 的设计视图，配置 ListView 控件，布局选择【网格】，样式选择【专业型】，同时选中【启用编辑】、【启用插入】、【启用删除】和【启用分页】复选框。单击【确定】按钮关闭该对话框。

(6) 切换至源视图，删除 ListView 控件所有模板中显示<Class>的列，修改 LayoutTemplate 模板中的列标题为中文描述。

(7) 定位到 InsertItemTemplate 标记，为了让用户选择“性别”，需要用 RadioButton 控件取代用于 Sgender 属性的 TextBox，修改后的代码如下：

```
<asp:RadioButton ID="RadioButton1" runat="server" Checked="True" Text="男" GroupName="gender" />
<asp:RadioButton ID="RadioButton2" runat="server" GroupName="gender" Text="女" />
```

(8) 在设计视图中选择 EntityDataSource1，打开其【属性】面板，切换到【事件】选项卡。双击 Inserting 事件，如图 6-15 所示。

(9) 在数据源控件的 Inserting 事件处理程序中，添加如下代码：

```
protected void EntityDataSource1_Inserting(object sender, EntityDataSourceChangingEventArgs e)
{
    Student stu = (Student)e.Entity;
    RadioButton rdo = (RadioButton)ListView1.InsertItem.FindControl("RadioButton1");
    if (rdo.Checked)
        stu.Sgender = "M";
    else
        stu.Sgender = "F";
}
```

上述代码在用户按下【插入】按钮时触发，在这一事件处理程序中，通过 e.Entity 获取当前记录，另外添加了一些代码来“发现”InsertItem 模板中的 RadioButton 控件。因为可能会有名称相同的多个控件(例如 InsertItemTemplate 和 EditItemTemplate 中的控件名可以相同)，所以不能直接访问 RadioButtonl 控件，而是需要在 InsertItem 对象上使用 FindControl 来搜索该控件。

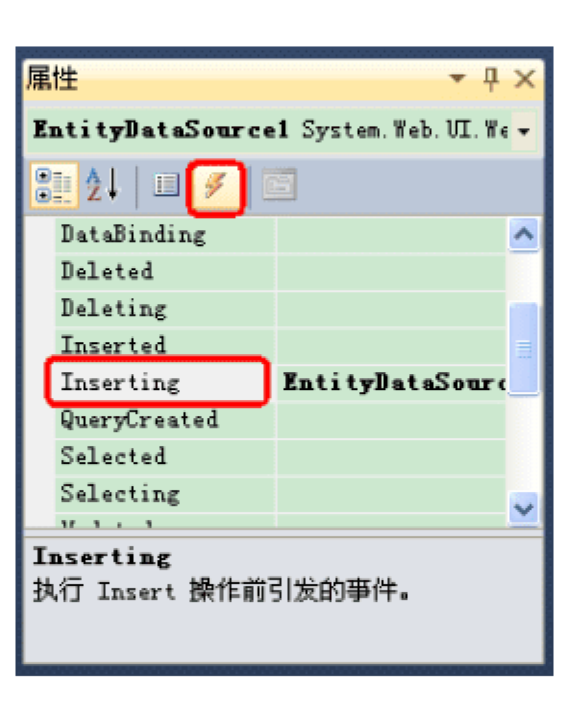


图 6-15 添加数据源控件的 Inserting 事件

(10) 编译并运行程序，在浏览器中打开 Default.aspx 页面，运行效果如图 6-16 所示。

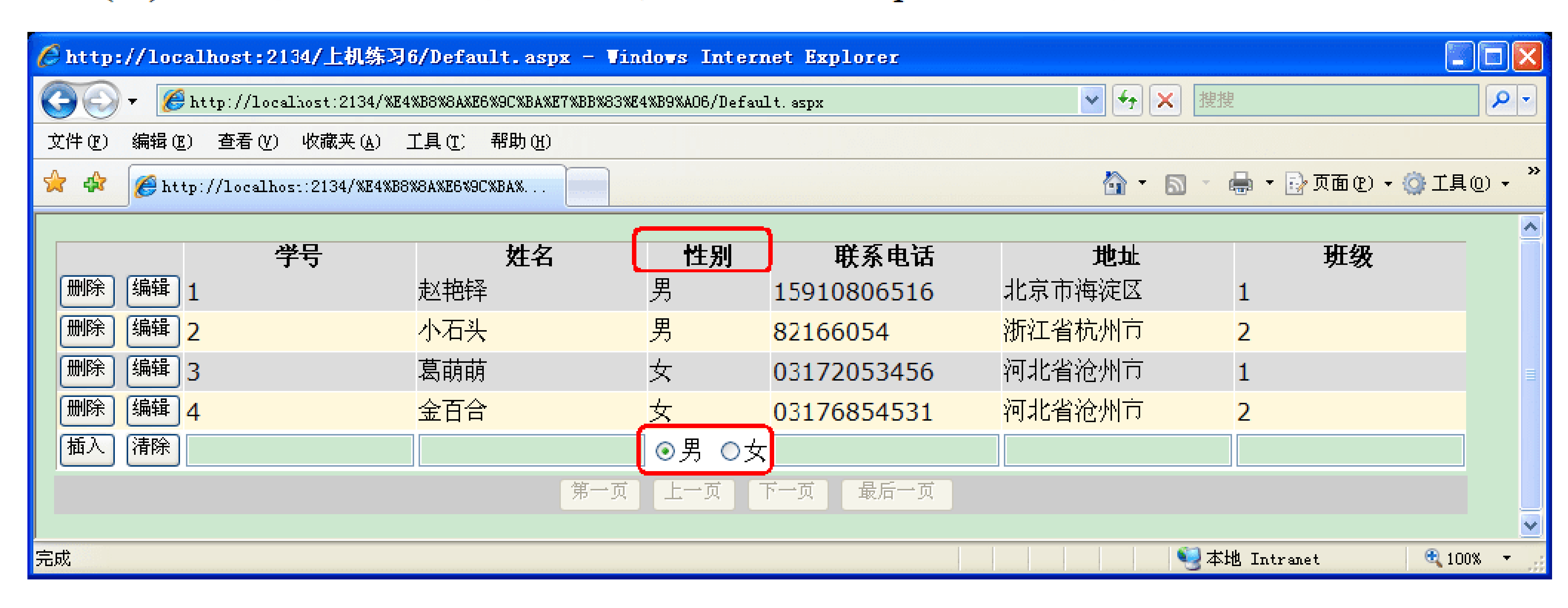
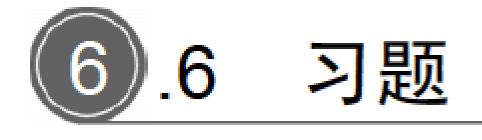
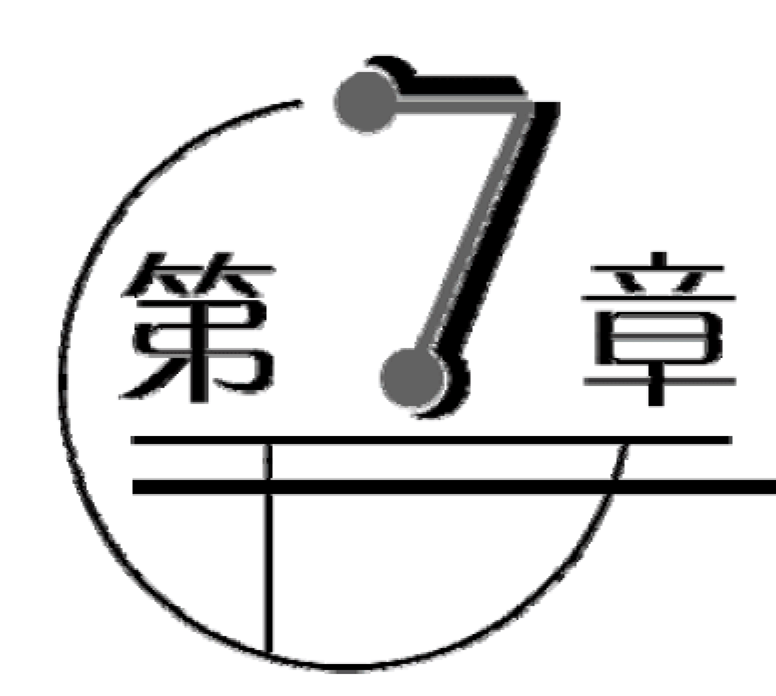


图 6-16 页面运行效果



6.6 习题

1. 要使用 LINQ，必须引入哪个命名空间？
2. 什么是匿名类，如何定义匿名类？
3. LINQ 查询表达式以什么开头？
4. 如何添加 ADO.NET 实体数据模型。
5. EntityDataSource 数据源控件有什么用？
6. 与其他数据控件(如 GridView 和 Repeater)相比，ListView 控件的主要优势在哪里？
7. 在上机练习的基础上，编辑 EditItemTemplate 模板，使得编辑记录时，用户也可以选择“性别”。



ASP.NET AJAX

学习目标

AJAX 是一种广泛而且十分有趣的技术，可以给站点增加许多功能。ASP.NET AJAX 采用异步编程方式，与前面学习的同步编程模式有所不同，最大的特点是提供对客户端脚本的自动管理。利用 ASP.NET AJAX 服务器控件，可以实现局部页面更新的效果。本章首先介绍了 ASP.NET AJAX 的基本知识，然后详细讲解 ASP.NET AJAX 服务器控件的使用方法，通过本章的学习，读者可掌握 ASP.NET AJAX 提供的各种服务器控件的用法，以及使用客户端 ASP.NET AJAX Library。

本章重点

- 了解 ASP.NET AJAX 的基本知识
- 使用 UpdatePanel 控件
- 使用 UpdateProgress 控件通知用户 Ajax 操作的进程
- 使用客户端 ASP.NET AJAX Library

7.1 AJAX 简介

本书前面已经介绍了浏览器如何与服务器进行交互。浏览器使用 GET 或 POST 方法来请求页面，服务器处理该页面，并回发生成的 HTML 代码。然后，浏览器解析该 HTML 代码并将页面呈现给用户，并有选择地下载任意的外部资源，如图像、脚本文件和 CSS 样式表。当用户之后与页面交互时(例如，通过单击按钮来提交一个已填好信息的联系表单)，页面被回发给服务器，之后浏览器会再次加载整个页面。

尽管上述模型已经在 Web 页面中使用了很多年，但是它仍然存在一些缺陷。首先，因为整个页面是在回发后被加载的，因此发送到浏览器的 HTML 代码量要远大于浏览器所需要的；加



载整个页面的第二个缺陷与浏览器显示页面的方式有关，由于整个页面被替换掉，因此浏览器会不得不关闭旧页面，再打开新页面，这样就会使页面“闪烁”，从而使其失去对用户的吸引力。Ajax 就是为了解决上述两个问题而产生的。

Ajax(Asynchronous JavaScript and XML)技术是由 Jesse James Garrett 提出的，是综合异步通信、JavaScript 以及 XML 等多种网络技术新的编程方式。如果从用户看到的实际效果来看，也可以形象地称之为无页面刷新技术。

Ajax 技术的主要内容包括：基于 Web 标准 XHTML+CSS 的表示；使用 DOM(Document Object Model)进行动态显示及交互；使用 XML 和 XSLT 进行数据交换及相关操作；使用 XMLHttpRequest 进行异步数据检索；使用 JavaScript 将所有的东西绑定在一起等。

Ajax 技术的最大优点是能在不更新整个页面的前提下维护数据。这使得 Web 应用程序能够更为迅捷地回应用户动作，并避免了在网络上发送那些没有改变过的信息。

2005 年，Microsoft 公司在专业开发人员大会上宣布将在 ASP.NET 上实现 Ajax 功能(开发代号为 Atlas)，主要是为了充分利用客户端 JavaScript、DHTML 和 XMLHttpRequest 对象，目的是帮助开发人员创建更具交互性的支持 Ajax 的 Web 应用程序。直到 2007 年 1 月，Microsoft 公司才真正推出了具有 Ajax 风格的异步编程模型，这就是 ASP.NET AJAX 1.0。同时，为了与其他的 Ajax 技术区分，Microsoft 公司用大写的 AJAX，并在其前面加上 ASP.NET。

ASP.NET AJAX 1.0 是以可以在 ASP.NET 2.0 之上安装的单独一个下载的形式发布的。从.NET Framework 3.5 开始，所有这些特性都成为 ASP.NET 所固有的，这意味着在构建或部署应用时，不再需要下载和安装单独的 ASP.NET AJAX 安装文件。

在 ASP.NET 4.0 中，它被完全集成在.NET 4.0 Framework 和 VWD 2010 中，并且与其他客户端架构(包括 jQuery)具有很好的互操作性。

通过 ASP.NET AJAX，开发人员可以实现如下功能：

- 创建无闪烁页面，它们允许刷新部分页面，而不需要全部重载，也不会影响页面的其他部分。
- 在这些页面刷新过程中给用户提供反馈。
- 更新部分页面，使用计时器按计划调用服务器端的代码。
- 访问服务器端 Web 服务和页面方法，使用它们返回的数据。
- 使用富客户端编程架构访问和修改页面中的元素，访问代码模型和典型系统。

ASP.NET AJAX 包括两个重要的部分：ASP.NET AJAX 服务器控件和客户端 ASP.NET AJAX Library。

对于 Web 开发来说，ASP.NET AJAX 从基础框架实现，到客户端与服务器的通信，都发生了翻天覆地的变化。相对于 ASP.NET 来说，ASP.NET AJAX 是一种更为成熟的 Web 开发技术。下面将介绍 ASP.NET AJAX 主要控件 ScriptManager、UpdatePanel、UpdateProgress 和 Timer 的使用方法。





7.2 使用 AJAX 控件

ASP.NET AJAX 完全综合集成到了 ASP.NET 和 VWD 中，这就意味着开发人员可以使用它了。对于在 VWD 中创建的每个新的 ASP.NET 4.0 Web 站点来说，它们的 AJAX 功能都已经被激活了。此外，工具箱中包含一个【AJAX Extensions】类别，该类别中包含了要在页面中使用的与 AJAX 相关的控件。VWD 还对 ASP.NET AJAX 提供了大力支持，它为服务器端的控件和客户端的 JavaScript 代码提供了智能感知功能。

7.2.1 ScriptManager 控件

ScriptManager 控件是 ASP.NET AJAX 的核心，它提供处理页面上的所有 ASP.NET AJAX 控件(UpdatePanel、UpdateProgress 等)的支持，没有该控件的存在，其他 ASP.NET AJAX 控件就无法工作，并且所有需要支持 ASP.NET AJAX 的 ASP.NET 页面上只能有一个 ScriptManager 控件。另外，ScriptManager 控件还可以生成相关的客户端代理脚本，以便能够在客户端脚本中访问 Web 服务。



1. ScriptManager 控件的属性和方法

ScriptManager 类有许多属性，其中大多数都用于高级场景。在很多情况下，不需要改变 ScriptManager 类的任何属性；而在有些情况下，需要改变或设置其某些属性。如表 7-1 所示列出了 ScriptManager 控件的一些常见属性。

表 7-1　ScriptManager 控件的重要属性

属　性	描　述
AllowCustomErrorsRedirect	该属性确定 AJAX 运行过程中出现的错误是否会导致加载自定义的错误页面。默认为 True；设置为 False 时，错误在浏览器中显示为 JavaScript 通知窗口，或者在禁止调试时对客户端隐藏。注意，如果没有配置任何自定义错误页面，错误就总是显示为 JavaScript 通知
AsyncPostBackErrorMessage	异步回传发生错误时的错误信息。如果没有使用自定义错误页面，这个属性允许自定义错误消息，当发生 AJAX 错误时，用户可以看到这条错误消息
AsyncPostBackTimeout	异步回传时超时限制，默认值为 90，单位为秒
EnablePageMethods	这个属性确定是否允许客户端代码调用页面内定义的方法。后面将讨论其工作原理
EnablePartialRendering	该属性确定 ScriptManager 是否支持使用 UpdatePanel 控件呈现部分页面。除非想阻止整个页面的部分更新，否则应该将它设置为 True
EnableCdn	若该属性设置为 True， ASP.NET 将会包含微软的 Content Delivery Network 站点上(而不是自己的服务器上)的客户端框架文件的链接。这样可以节省一些带宽，如果用户已经从使用这些文件的其他站点那里获取了这些文件的高速缓存副本的话，这样做还可能会提高页面首次加载时的速度



(续表)

属　性	描　述
MicrosoftAjaxMode	该属性可用于确定是否包含 Microsoft AJAX 客户端库。该属性允许使用 ScriptManager 控件完成与服务器相关的任务(如注册客户端脚本)，而不需要在页面中嵌入客户端框架
ScriptMode	指定 ScriptManager 发送到客户端的脚本的模式，有 4 种模式：Auto、Inherit、Debug、Release，默认值为 Auto
ScriptPath	设置所有的脚本块的根目录，作为全局属性，包括自定义的脚本块或者引用第三方的脚本块。如果在 Scripts 中的<asp:ScriptReference />标签中设置了 Path 属性，它将覆盖该属性
Scripts	ScriptManager 控件的<Scripts>子元素允许添加客户端在运行时必须下载的其他的 JavaScript 文件
CompositeScript	和<Scripts>元素一样，<CompositeScript>元素也允许添加其他的 JavaScript 文件。但是，在<Composite-Script>元素下注册的文件都被合并为一个单独的、可下载的文件，从而可以减小网络开销并提高页面的性能
Services	<Services>元素允许定义客户端页面能够访问的 Web 服务
OnAsyncPostBackError	异步回传发生异常时的服务端处理函数，在这里可以捕获一场信息并作相应的处理
OnResolveScriptReference	指定 ResolveScriptReference 事件的服务器端处理函数，在该函数中可以修改某一条脚本的相关信息如路径、版本等



ScriptManager 控件是客户端页面和服务器之间的桥梁。它管理脚本资源(客户端使用的 JavaScript 文件)，负责部分页面的更新(如前所述)，处理与 Web 站点的交互，例如 Web 服务和 ASP.NET 应用程序服务(如成员、角色和配置文件)。

如果认为只在一小部分页面上需要 AJAX 性能，那么通常可以将 ScriptManager 控件直接放置到内容页中。但是，也可以将 ScriptManager 控件放置在母版页中，这样它便在整个站点中都可用。

2. ScriptManager 控件的用法

要使用 ASP.NET AJAX 提供的功能，必须在网页中包含一个 ScriptManager 控件。添加 ScriptManager 控件后，将生成如下代码：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
```

在介绍完 UpdatePanel 控件后，将一起举例说明 ScriptManager 控件的用法。

7.2.2　UpdatePanel 控件

UpdatePanel 控件是 ASP.NET AJAX 中很重要的一个控件，它可以用来创建局部更新的 Web 应用程序。有了 UpdatePanel 控件，开发者不需要编写任何客户端脚本，只需在页面上添加 UpdatePanel 控件和 ScriptManager 控件就可以自动实现局部更新。

1. UpdatePanel 控件的工作原理

UpdatePanel 控件的工作过程如图 7-1 所示。

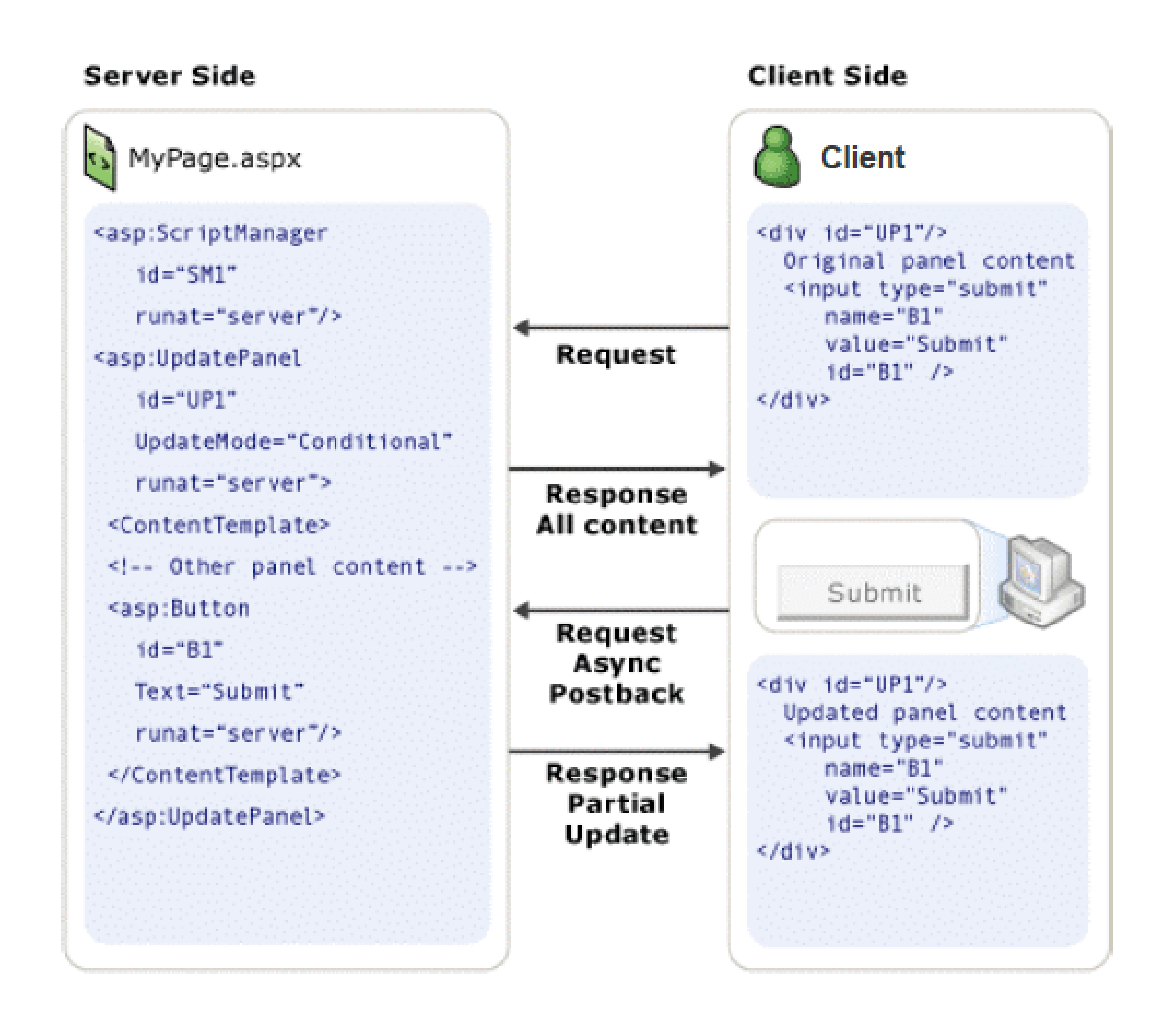


图 7-1　UpdatePanel 控件的工作原理

知识点

UpdatePanel 控件的工作依赖于 ScriptManager 控件和客户端 PageRequestManager 类，当 ScriptManager 允许页面局部更新时，它会以异步的方式回传给服务器，与传统的整页回传方式不同的是只有包含在 UpdatePanel 中的页面部分才会被更新，在从服务器返回 XHTML 之后，PageRequestManager 会通过操作 DOM 对象来替换需要更新的代码片段。

当客户端第一次向服务器发出请求时，服务器返回整个页面。除此之外，均通过异步回传方式对页面进行局部更新。

2. UpdatePanel 控件的属性

UpdatePanel 控件的常用属性如表 7-2 所示。

表 7-2　UpdatePanel 控件的重要属性

属　　性	描　　述
ChildrenAsTriggers	该属性确定位于 UpdatePanel 内的控件能否刷新 UpdatePanel。其默认值是 True，当该值设置为 False 时，必须将 UpdateMode 设置为 Conditional。注意，当设置为 False 时，UpdatePanel 内定义的控件仍然会引发到服务器的回发，只是不再自动更新面板
Triggers	Triggers 集合包含 PostBackTrigger 和 AsyncPostBackTrigger 元素。如果要实现完整的页面刷新，那么就可以用第一个；而如果要使用在面板之外定义的控件更新 UpdatePanel，则用第二个



(续表)

属　性	描　述
RenderMode	该属性表示 UpdatePanel 最终呈现的 HTML 元素。Block(默认)表示<div>，Inline 表示<span>
UpdateMode	该属性表示 UpdatePanel 的更新模式，有两个选项：Always 和 Conditional。Always 是不管有没有 Trigger，其他控件都将更新该 UpdatePanel，Conditional 表示只有当前 UpdatePanel 的 Trigger，或 ChildrenAsTriggers 属性为 true 时当前 UpdatePanel 中控件引发的异步回送或者整页回送，或是服务器端调用 Update()方法才会引发更新该 UpdatePanel
ContentTemplate	尽管在 UpdatePanel 的【属性】面板中不可见，但<ContentTemplate>是 UpdatePanel 的一个重要属性。它是一个容器，用于定义 UpdatePanel 的内容，可以将控件放置在该容器中作为 UpdatePanel 的子控件。如果忘记了这个必需的 ContentTemplate 属性，VWD 会发出一条警告

3. 实现局部更新

在一个页面中，如果需要局部更新的内容较少，可以放置一个 UpdatePanel 控件，在该控件内实现局部更新的效果。下面通过具体的实例介绍在 UpdatePanel 中实现局部更新的方法。

【例 7-1】在 UpdatePanel 中实现局部更新。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 7-1】。

(2) 在 Default.aspx 的设计视图中，添加一个 ScriptManager，一个 UpdatePanel 控件。

(3) 在 UpdatePanel 内部添加一个 Label 控件和两个 Button 控件，同时在 UpdatePanel 外面也添加两个 Button 控件。

(4) 设置 4 个按钮的 Text 属性分别为“UpdatePanel 内的完整页面刷新”、“异步刷新”、“UpdatePanel 外”和“UpdatePanel 外异步刷新”。

(5) 选中 UpdatePanel 控件，通过【属性】面板设置 Triggers 属性，这是一个集合属性，单击属性右侧的按钮，打开【UpdatePanelTrigger 集合编辑器】对话框，单击【添加】按钮右侧的倒三角形，打开一个下拉菜单，如图 7-2 所示，可以添加两类元素：PostBackTrigger 和 AsyncPostBackTrigger。本例中设置 Button1 为 PostBackTrigger，Button4 为 AsyncPostBackTrigger。

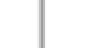


提示

如果在【属性】面板中找不到 Triggers 属性，则可能是因为页面中没有添加 ScriptManager 控件。

(6) 切换到源视图，相应的代码如下：

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
```

```
                <asp:Label ID="Label1" runat="server" Text="获取当前时间"></asp:Label>
                <br />
                <asp:Button ID="Button1" runat="server" Text="UpdatePanel 内的完整页面刷新"
                    onclick="Button1_Click" />
                <asp:Button ID="Button2" runat="server" Text="异步刷新" onclick="Button2_Click"
                    style="height: 21px" />
            </ContentTemplate>
            <Triggers>
                <asp:PostBackTrigger ControlID="Button1" />
                <asp:AsyncPostBackTrigger ControlID="Button4" EventName="Click" />
            </Triggers>
        </asp:UpdatePanel><br />
        <asp:Button ID="Button3" runat="server" Text="UpdatePanel 外"
            onclick="Button3_Click" />
        <asp:Button ID="Button4" runat="server" onclick="Button4_Click"
            Text="UpdatePanel 外异步刷新" />
</div>
```

(7) 分别为 4 个 Button 控件添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 内整个页面刷新，当前时间：" + DateTime.Now.ToString();
}
protected void Button2_Click(object sender, EventArgs e)
{
    Label1.Text = "局部刷新无闪烁，当前时间：" + DateTime.Now.ToString();
}
protected void Button3_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 外整个页面刷新，当前时间：" + DateTime.Now.ToString();
}
protected void Button4_Click(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel 外异步刷新，当前时间：" + DateTime.Now.ToString();
}
```

(8) 按【Ctrl + F5】组合键运行程序，分别单击不同的按钮，观察有什么不同，如图 7-3 所示是局部刷新无闪烁的情况。

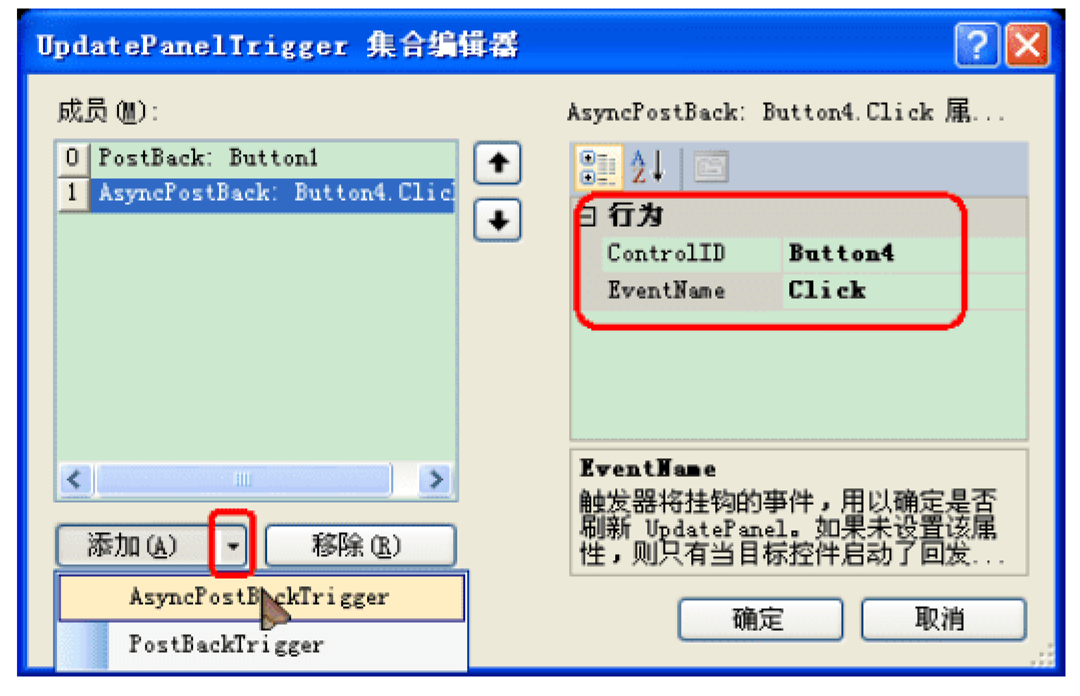


图 7-2　【UpdatePanelTrigger 集合编辑器】对话框

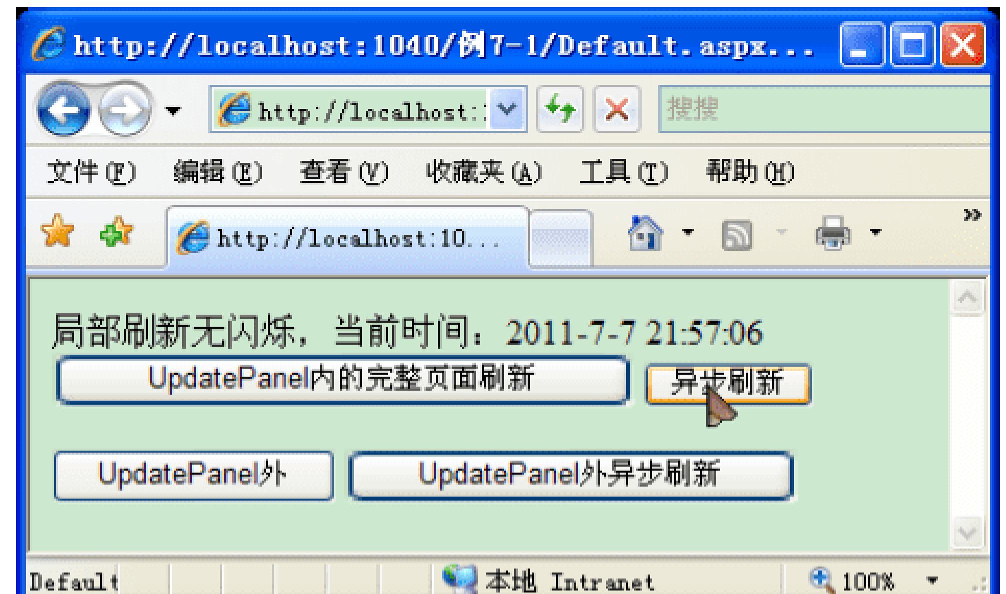


图 7-3　页面运行效果图

知识点

通过 Triggers 属性可以将 UpdatePanel 外的控件设置为异步刷新，所以在使用母版页的时候，通常可以在母版页中放置 ScriptManager 控件；在内容页使用 UpdatePanel，通过 Triggers 属性设置母版页中的控件实现局部更新。

可以看到，虽然单击每个按钮都实现了获取最新的时间，但页面刷新效果却不同。通常默认情况下，在 UpdatePanel 内部的服务器控件采用的是异步回传方式，实现 UpdatePanel 的局部更新，而在 UpdatePanel 外面的服务器控件采用的是同步回传方式，实现整个页面的刷新。而使用了 Triggers 属性后，虽然 Button1 按钮在 UpdatePanel 内部，但实现的是整个页面的更新，而在 UpdatePanel 外面的 Button4 按钮却实现了 UpdatePanel 局部更新。

4. 在同一页面上使用多个 UpdatePanel

使用 UpdatePanel 的时候并没有限制在一个页面中使用多少个 UpdatePanel，所以可以为不同的区域加上不同的 UpdatePanel。由于 UpdatePanel 默认的 UpdateMode 是 Always，如果页面上有一个局部更新被触发，则所有的 UpdatePanel 都将更新，要想只更新某个 UpdatePanel，只需把 UpdateMode 设置为 Conditional 即可。

下面的【例 7-2】就包括两个 UpdatePanel，其中一个用来输入数据，而另一个则用来显示数据，两个 UpdatePanel 的 UpdateMode 属性都设置为 Conditional，当单击【新增】按钮时，两个 UpdatePanel 都更新，单击【取消】按钮时，只有 UpdatePanel2 更新。

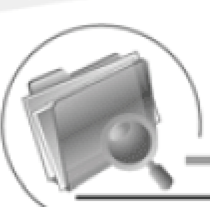
【例 7-2】在同一页面中使用多个 UpdatePanel。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 7-2】。

(2) 在 Default.aspx 的设计视图中，添加一个 ScriptManager 和两个 UpdatePanel 控件。

(3) 在 UpdatePanel1 内部添加 ListBox 控件和一个 Label 控件，在 UpdatePanel2 中放置一个 TextBox 控件、一个 Label 控件和两个 Button 控件。

(4) 设置两个 UpdatePanel 控件的 UpdateMode 属性为 Conditional。设置 UpdatePanel1 的 Triggers 属性为 Button1 异步刷新 AsyncPostBackTrigger。



(5) 切换到源视图，修改相应的代码如下所示：

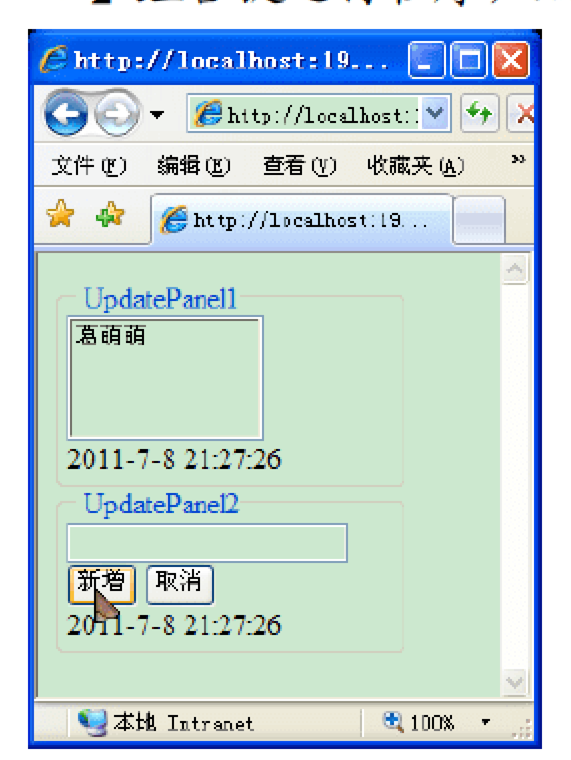
```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
        <ContentTemplate>
            <fieldset style="width:180px;">
            <legend >UpdatePanel1</legend>
            <asp:ListBox ID="ListBox1" runat="server" Width="108px"></asp:ListBox>
            <br />
            <asp:Label ID="Label1" runat="server" ><%=DateTime.Now %></asp:Label>
        </ContentTemplate>
        <Triggers>
            <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
        </Triggers>
    </asp:UpdatePanel>
    <asp:UpdatePanel ID="UpdatePanel2" runat="server">
        <ContentTemplate>
            <fieldset style="width:180px;">
            <legend >UpdatePanel2</legend>
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
            <asp:Button ID="Button1" runat="server" Text="新增" onclick="Button1_Click" />
            <asp:Button ID="Button2" runat="server" Text="取消" onclick="Button2_Click" />
            <br />
            <asp:Label ID="Label2" runat="server" ><%=DateTime.Now %></asp:Label>
        </ContentTemplate>
    </asp:UpdatePanel>
</div>
```

(6) 添加两个 Button 控件的事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    ListBox1.Items.Add(TextBox1.Text);
    TextBox1.Text = "";
}
protected void Button2_Click(object sender, EventArgs e)
{
    TextBox1.Text = "";
}
```



(7) 按【Ctrl + F5】组合键运行程序，效果如图 7-4 所示。



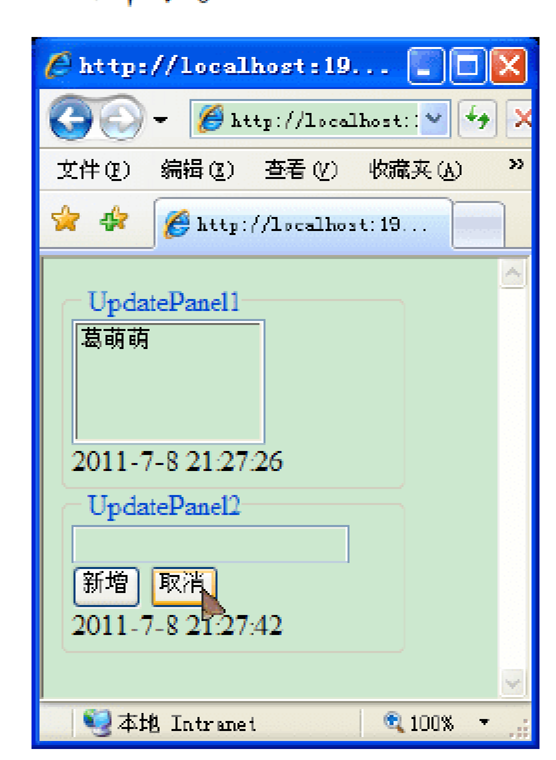


图 7-4　多个 UpdatePanel 控件的运行效果

当单击【新增】按钮时，将更新 UpdatePanel1 和 UpdatePanel2，而单击【取消】按钮时，只更新 UpdatePanel2。



提示

当发生 UpdatePanel 控件异步更新错误时，默认情况下会弹出一个错误对话框。如果设计者觉得不符合用户习惯，可以通过 ScriptManager 控件的 OnAsyncPostBackError 事件和 AsyncPostBackErrorMessage 属性捕捉和设置回传时的错误信息。

7.2.3　UpdateProgress 控件

虽然使用 UpdatePanel 和 ScriptManager 已经足以创建无闪烁页面，但 ASP.NET AJAX 提供了更多控件来增强用户在启用了 AJAX 的 Web 站点中的体验。改进用户体验的方法之一是使用 UpdateProgress 控件，另一种选择是使用 Timer 控件。本节就将介绍 UpdateProgress 控件。

UpdateProgress 控件一般与 UpdatePanel 控件联合使用，即在 UpdatePanel 异步更新过程中，显示提示信息。这些信息可以是一段文字、进度条或各种动画。当异步更新完成时，提示信息自动消失。

1. UpdateProgress 控件的属性

UpdateProgress 控件的常用属性如表 7-3 所示。

表 7-3　UpdateProgress 控件的常用属性

属　性	描　述
AssociateUpdatePanelID	设置哪个 UpdatePanel 控件产生的回送会显示 UpdateProgress 的内容，当关联的 UpdatePanel 控件忙于刷新时，就会显示在<ProgressTemplate>元素中定义的内容。通常要在模板中放入文本或动画图像(也接受其他标记)来让用户知道正在发生的事情

(续表)

属　性	描　述
DisplayAfter	当引发回送后多少毫秒会显示 UpdateProgress 控件的内容，默认值是 500 毫秒
DynamicLayout	设置 UpdateProgress 控件的显示方式。如果为 true，当 UpdateProgress 控件不显示的时候不占用空间(默认)；为 false，当 UpdateProgress 控件不显示的时候仍然占用空间
ProgressTemplate	获取或设置定义 UpdateProgress 控件内存的模板

如果没有设定 UpdateProgress 控件的 AssociateUpdatePanelID 属性，则任何一个异步更新都会使 UpdateProgress 控件显示出来。相反，如果将 UpdateProgress 控件的 AssociateUpdatePanelID 属性设置为某个 UpdatePanel 控件的 ID，那么，只有该 UpdatePanel 控件引发的异步更新才会使相关联的 UpdateProgress 控件显示出来。



知识点

必须为 UpdateProgress 控件定义模板。否则，在 UpdateProgress 控件的 Init 事件发生期间会触发异常。可通过将标记添加到 ProgressTemplate 元素，以声明的方式指定 ProgressTemplate 属性。如果要动态创建 UpdateProgress 控件，则应在页面的 PreRender 事件发生期间或发生之前进行创建。

2. 使用 UpdateProgress 控件

下面的【例 7-3】将演示 UpdateProgress 控件的用法，当 UpdatePanel 控件异步更新时，显示 UpdateProgress 控件的提示内容。

【例 7-3】使用 UpdateProgress 控件给用户提供反馈信息。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 7-3】。

(2) 在【解决方案资源管理器】面板中，新建一个 images 文件夹，然后添加进度条动画文件 progress.gif。

(3) 在 Default.aspx 的设计视图中，添加一个 ScriptManager、一个 UpdatePanel 控件和一个 UpdateProgress 控件。

(4) 在 UpdatePanel 控件中添加一个 Label 控件和一个 Button 控件，设置 Button 控件的 Text 属性为“提交”。

(5) 在 UpdateProgress 控件中添加文本“正在刷新，请稍候...”和一个 Image 控件，Image 控件的 ImageUrl 属性指向前面的进度条动画图片 progress.gif。

(6) 切换到源视图，生成的代码如下所示：

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server" >
```



```
        </asp:ScriptManager>
        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
                <br />
                <asp:Button ID="Button1" runat="server" Text="提交" onclick="Button1_Click" />
            </ContentTemplate>
        </asp:UpdatePanel>
        <asp:UpdateProgress ID="UpdateProgress1" runat="server" AssociatedUpdatePanelID="UpdatePanel1">
            <ProgressTemplate>
                正在刷新，请稍候...<asp:Image ID="Image1" runat="server" ImageUrl="~/images/progress.gif" />
            </ProgressTemplate>
        </asp:UpdateProgress>
</div>
```



(7) 添加按钮控件的单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(5000);   //等待 5 秒
    Label1.Text = DateTime.Now.ToString();
}
```

(8) 编译并运行程序，运行效果如图 7-5 所示。

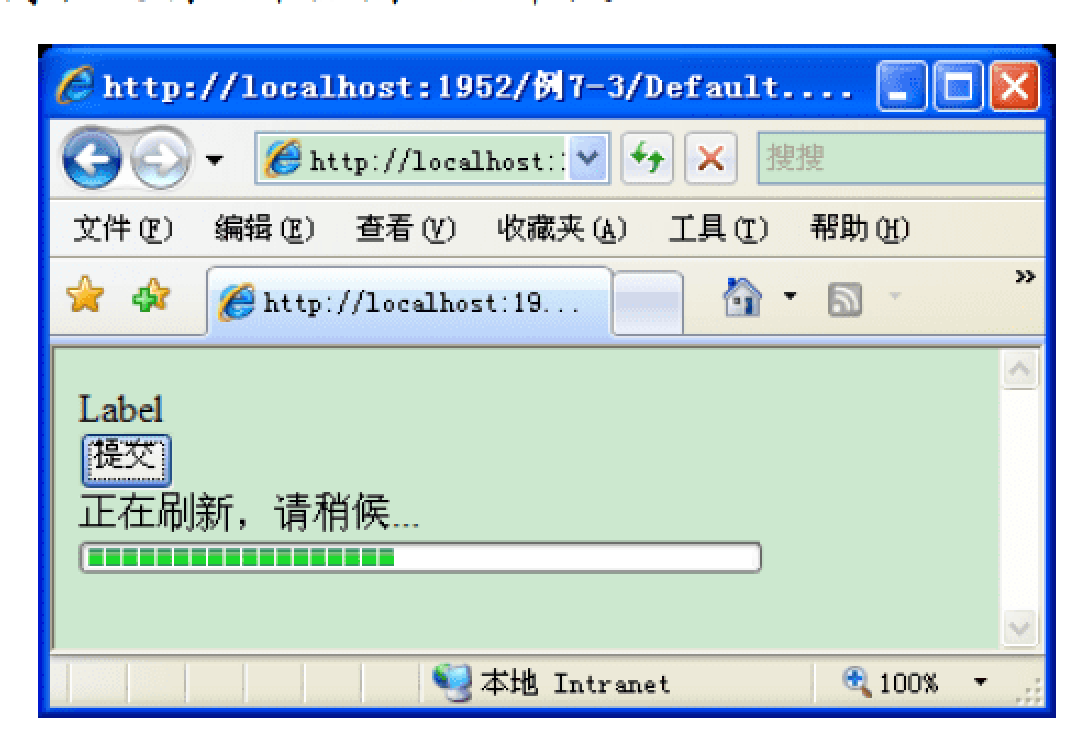


图 7-5　使用进度提示控件

本例中只有一个 UpdateProgress 控件，也可以在一个页面中使用多个 UpdateProgress 控件，通过设置 AssociateUpdatePanelID 属性来指点相关联的 UpdatePanel 控件。



提示

虽然一个页面允许有多个 UpdateProgress 控件，但是在实际中，一般在一个页面中只放置一个 UpdateProgress 控件。

7.2.4　Timer 控件

Timer 控件是 ASP.NET AJAX 中又一个重要的服务器控件。通过它可以完成局部页面的定时更新，从而实现图片自动播放、超时自动退出等功能。

1. 属性和事件

Timer 控件的常用属性和事件如表 7-4 所示。



表 7-4　Timer 控件的常用属性和事件

属性和事件	描　述
Interval	该属性用于指定间隔时间，其设置值的单位是毫秒，默认值则是 60000 毫秒
Enabled	该属性用于表示是否允许 Tick 事件
Tick	该事件在 Interval 指定的间隔到期后触发

需要注意的是，将 Timer 控件的 Interval 属性设置为较小的值会使得回送频率增加，也很容易使得 Web 服务器的流量大增，对整体资源耗用与效率都会造成不良的影响。因此尽量在确实需要的时候才使用 Timer 控件来定时更新页面上的内容。

Timer 控件在 UpdatePanel 控件的内外是有区别的。当 Timer 控件在 UpdatePanel 控件内部时，JavaScript 计时组件只有在一次回传完成后才会重新建立。也就是说直到网页回传完成之前，定时器间隔时间不会从头计算。例如，设置 Timer 控件的 Interval 属性值为 3000ms(3 秒)，但是回传操作本身却花了 2 秒才完成，则下一次的回传将发生在前一次回传被引发之后的 5 秒。而如果 Timer 控件位于 UpdatePanel 控件之外，则当定时器间隔到期以后，定时器间隔时间会立刻重新计算，下一次回传将发生在前一次回传被引发之后的 3 秒。

2. 使用 Timer 控件定时更新 UpdatePanel

Timer 控件的用法非常简单。该控件按照指定的时间间隔激活其 Tick 事件。在这个事件处理程序中，添加要刷新页面的代码即可。

下面的【例 7-4】是一个在 UpdatePanel 内部使用 Timer 控件的简单示例，该示例实现图片的自动刷新。



【例 7-4】在 UpdatePanel 内部使用 Timer 控件。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 7-4】。

(2) 在【解决方案资源管理器】面板中，新建一个 images 文件夹，然后添加需要循环显示的图片文件 Club1.gif、Club2.gif……Club13.gif。

(3) 在 Default.aspx 的设计视图中，添加一个 ScriptManager 和一个 UpdatePanel 控件。

(4) 在 UpdatePanel 控件中添加一个 Timer 控件和一个 Image 控件，设置 Timer 控件的 Interval 属性为 2000，设置 Image 控件的 ImageUrl 属性为 images 目录下的 Club1.gif。

(5) 此时源视图中生成的代码如下：

```
<div>
<h3>使用 Timer 控件循环显示梅花 1—K</h3>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <asp:Timer ID="Timer1" runat="server" Interval="2000" ontick="Timer1_Tick">
            </asp:Timer>
            <asp:Image ID="Image1" runat="server" ImageUrl="~/images/Club1.gif" />
        </ContentTemplate>
    </asp:UpdatePanel>
</div>
```

(6) 添加页面的 Load 事件处理程序和 Timer 控件的 Tick 事件处理程序，代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Page.IsPostBack == false)
    {
        ViewState["count"] = 1;//设置网页上的变量
    }
}
protected void Timer1_Tick(object sender, EventArgs e)
{
    ViewState["count"] = (int)ViewState["count"] % 13 + 1;
    Image1.ImageUrl = string.Format("~/images/Club{0}.gif", ViewState["count"]);
}
```

上述代码中使用到了前面学习过的视图状态来存放一个计数变量。



(7) 按【Ctrl + F5】组合键运行程序，可以看到每间隔 2 秒显示一张扑克牌，循环显示梅花 1—K，如图 7-6 所示。

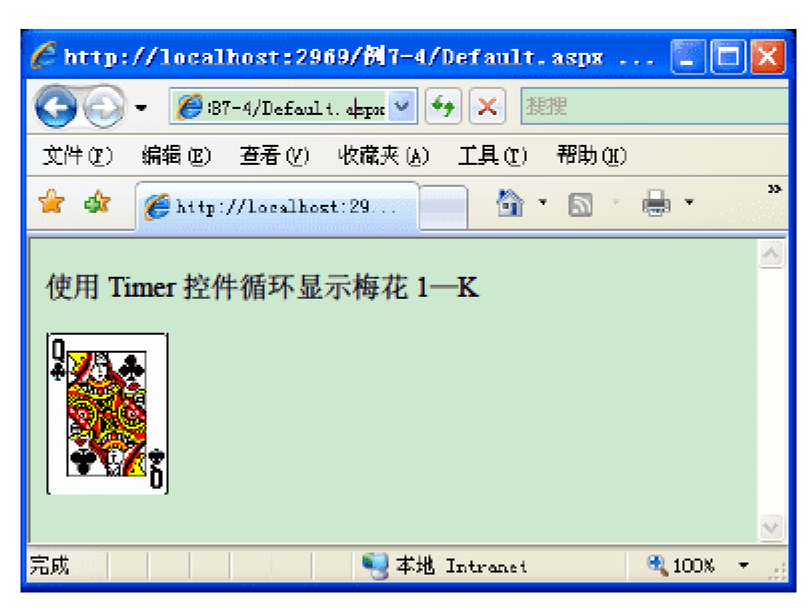


图 7-6 使用 Timer 控件定时刷新页面

7.2.5 ScriptManagerProxy 控件



ScriptManagerProxy 控件是内容页面与母版页中定义的 ScriptManager 控件之间的桥梁。在页面中，控件 ScriptManagerProxy 的外观和操作与标准控件 ScriptManager 很相似。但是，ScriptManagerProxy 控件实际上只是一个 proxy 类，该类可以将其所有的设置传递给母版页中真正的 ScriptManager 控件。

在本书第 8 章介绍 Web 服务的时候会用到该控件，此处不做过多介绍。

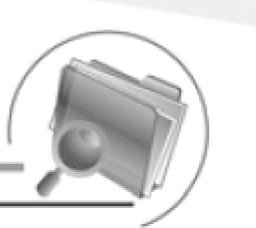
ASP.NET AJAX 包含的内容还有很多，在此无法一一介绍。ASP.NET AJAX 的服务器和客户机端部分可能是最大的、使用最多的功能，其他的功能也都是比较实用的，例如，ASP.NET AJAX 控件工具箱，它是一个非常好的扩展控件工具包，带有诸如日历扩展器和能自动完成的文本框这样的功能。包括 40 多个免费的扩展控件，而且一直都在增加，可以在网站 http://www.asp.net/ajax/AjaxControlToolkit/Samples 上查看和下载控件工具箱。

7.3 客户端 ASP.NET AJAX Library

客户端 ASP.NET AJAX Library 的功能非常强大，并且能够提供浏览器中纯 JavaScript 所缺少的许多功能。Microsoft 从.NET Framework 中吸取了许多好的功能，然后将它们转换为客户端 JavaScript 对应的功能。这就意味着可以在支持 JavaScript 的 Web 站点中使用熟悉的.NET 概念，即使 JavaScript 不支持这些概念。

客户端库包括 6 个顶级的命名空间(包括根命名空间 Sys)和一个全局命名空间，它们允许访问 30 多个类，而这些类具有数百种有用的方法，可用来帮助创建富客户端 Web 界面。

例如，可以使用客户端框架建立和处理页面事件(如 load 和 unload)，使用 WebRequest 类对其他的 Web 页面发出请求，根据 CultureInfo 类的代码中客户端的文化背景设置来呈现不同格式的数据，执行数据绑定(绑定的数据源有多种，如可以将 Web 服务绑定到用户的界面控件中，从



而显示并编辑它们)，以及其他情况。

全局命名空间中包含了一些成员和类型，它们扩展了 JavaScript 原先设计的特性。还包含了在 JavaScript 中发现的类型，如 Array、Boolean、Error 、Number、 Object 和 String，这些类型已经被扩展为包含模仿.NET Framework 的行为。如表 7-5 所示列出了这些类型的一些常用的方法。

表 7-5　全局命名空间中的成员和类型的常用方法

类　型	方　法	用　途
String	format	使用指定的格式字符串和参数返回一个格式化的字符串，例如： var str = String.format('Hello {0}', name);
	endsWith startsWith	用于确定一个字符串是否以另一个字符串作为开始或结束，例如： var isGemm = 'Ge Mengmeng'.startsWith('Ge');
	trim trimEnd trimStart	删除字符串前面和后面的空格，例如： var trimmed = '　　Zhao Yanduo　　'.trim();// results in Zhao Yanduo
Boolean	parse	将保留包含有文本 true or 或 false 的字符串转换成真正的布尔值，出现其他值时则抛出一个异常，例如： var isTrue = Boolean.parse('true');
Date	format	可以获得日期的不同的格式化字符串表示，例如： alert(new Date().format('f'));



客户端 ASP.NET AJAX Library 很大，所以并不总是能很容易地找到需要的方法、类或命名空间。为了更有效地使用客户端库，可以使用一些很好的资源。第一个就是“智能感知”，它可以在不同的类型中提供可用成员，它通过查找指定类型的值来完成这个工作。

例如，如图 7-7 所示，VWD 推测变量 name 的类型是字符串，这是因为该变量提供了与字符串相关的方法，如 trim 和 trimStart，这两个方法已经被客户端库添加到 String 类型中了。如果指定一个数字给变量，就会得到一个与数字类型对应的列表，如图 7-8 所示。

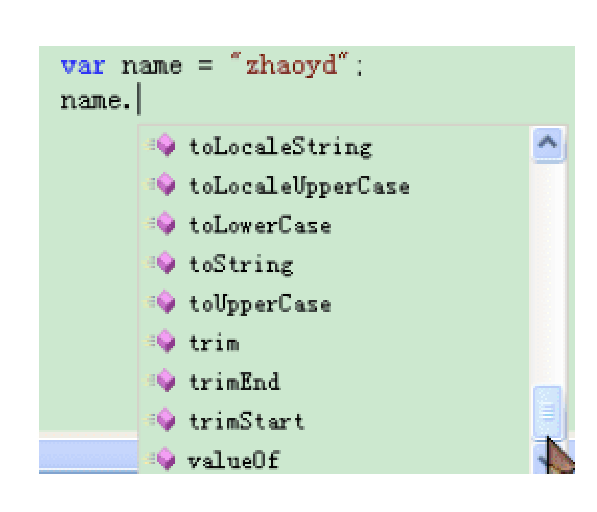


图 7-7　智能感知推测变量为字符串

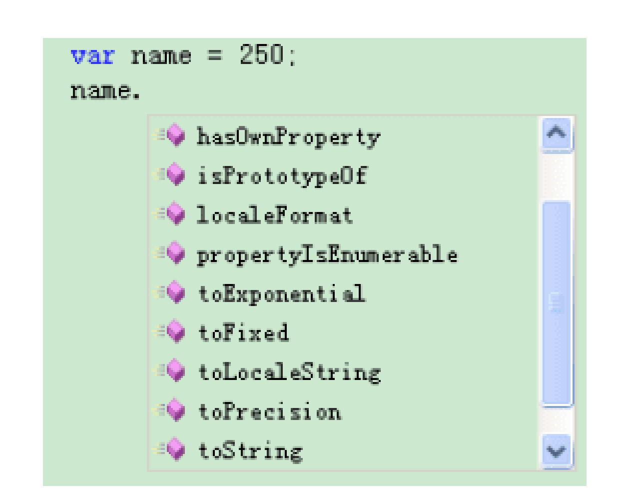


图 7-8　数字类型变量对应的方法列表

许多成员已经被文档化，因此可以得到一个很好的提示文本，来说明如何使用这些成员。如果没有看到正确的方法出现，则可能是页面(或母版页)中没有 ScriptManager 控件。没有该控件，客户端 JavaScript 架构框架将无法使用，因此 VWD 就会禁止使用浏览器的最终页面中不可用的功能。有时“智能感知”很可能并没有在列表中显示所期望的成员，但是这并不能说明这些

方法不可用。因为 JavaScript 不是一种强类型的语言，所以 VWD 需要做许多的解析、推断和猜测工作来提供正确的选项。

【例 7-5】客户端 ASP.NET AJAX Library 的一个简单应用。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 7-5】。

(2) 在 Default.aspx 的设计视图中，添加一个 ScriptManager。

(3) 在<ScriptManager>的结束标记下方，添加一个 Input (Text)和一个 Input (Button)控件，方法是从工具箱的 HTML 类别中拖动它们。通过使用纯 HTML 元素而不是 ASP.NET 服务器控件，可以看到要写的代码在客户端执行。将按钮的 value 设置为“提交”。相应的代码如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<input id="Text1" type="text" />
<input id="Button1" type="button" value="提交" />
```


(4) 在这两个 HTML 控件的下面，添加如下客户端 JavaScript 代码块：

```
<script type="text/javascript">
    $addHandler($get('Button1'), 'click', Hello);
    function Hello() {
        var name = $get('Text1').value;
        if (name.length == 0)
            alert("请输入姓名");
        else
            alert("Hello " + name);
    }
</script>
```

其中，$addHandler 是 AJAX 框架中定义的 Sys.UI.DomEvent 类的 addHandler 方法的一个快捷方式。可以用它在页面中注册这些对象的特定事件的事件处理程序。这与在 C#服务器端代码中看到的事件处理程序类似。$get 来获得对 HTML 标记的引用。


提示

$get 和$addHandler 最大的好处是可以在任何站点中使用它们。所需做的就是在母版页或内容页内包含 ScriptManager 控件，并准备使用客户端框架。

(5) 编译并运行程序，在文本框中输入一个名字，单击【提交】按钮，将弹出相应的欢迎对话框，如图 7-9 所示。

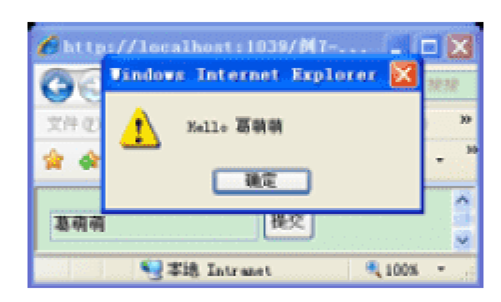


图 7-9　客户端 AJAX Library 应用示例



7.4　上机练习

本章的上机练习将演示 ASP.NET AJAX 控件的更多使用技巧，包括进度条控件 UpdateProgress 的取消功能和 Timber 控件定时更新多个 UpdatePanel。通过上机练习使读者进一步熟悉 ASP.NET AJAX 控件的用法和技巧。

7.4.1　进度条的取消功能

UpdateProgress 控件还支持另一个技术细节：即支持取消命令按钮。当用户单击了取消按钮时，异步回调将立即被终止，该 UpdateProgress 控件将消失，页面恢复到原来的状态。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 7-1】。

(2) 在【解决方案资源管理器】面板中，新建一个 images 文件夹，然后添加进度条动画文件 progress.gif。

(3) 在 Default.aspx 的设计视图中，添加一个 ScriptManager、一个 UpdatePanel 控件和一个 UpdateProgress 控件。

(4) 在 UpdatePanel 控件中添加一个 Label 控件和一个 Button 控件，设置 Button 控件的 Text 属性为“提交”，Label 控件的 Text 属性为“获取当前时间”。

(5) 在 UpdateProgress 控件中添加一个 Image 控件，其 ImageUrl 属性指向前面的进度条动画图片 progress.gif，然后再添加一个 HTML 控件 Input(Button)，设置按钮的 Text 属性为“取消”。

(6) 切换到源视图，在上述所有控件的下面添加如下客户端 JavaScript 代码块：

```
<script type="text/javascript">
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    $addHandler($get('Button2'), 'click', AbortPostBack);
    prm.add_initializeRequest(InitializeRequest);
    function InitializeRequest(sender, args) {
        if(prm.get_isInAsyncPostBack())
            args.set_cancel(true);
    }
    function AbortPostBack() {
        if (prm.get_isInAsyncPostBack())
            prm.abortPostBack();
    }
</script>
```

上述代码将“取消”按钮的 Click 事件连接到 AbortPostBack()方法，该方法用来取消当前的回发请求。

(7) 添加【提交】按钮控件的单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    System.Threading.Thread.Sleep(5000);   //等待 5 秒
    Label1.Text = DateTime.Now.ToString();
}
```

(8) 编译并运行程序，单击【提交】按钮，在回发过程中单击【取消】按钮，将取消回发，页面恢复到原来的状态，如图 7-10 所示。

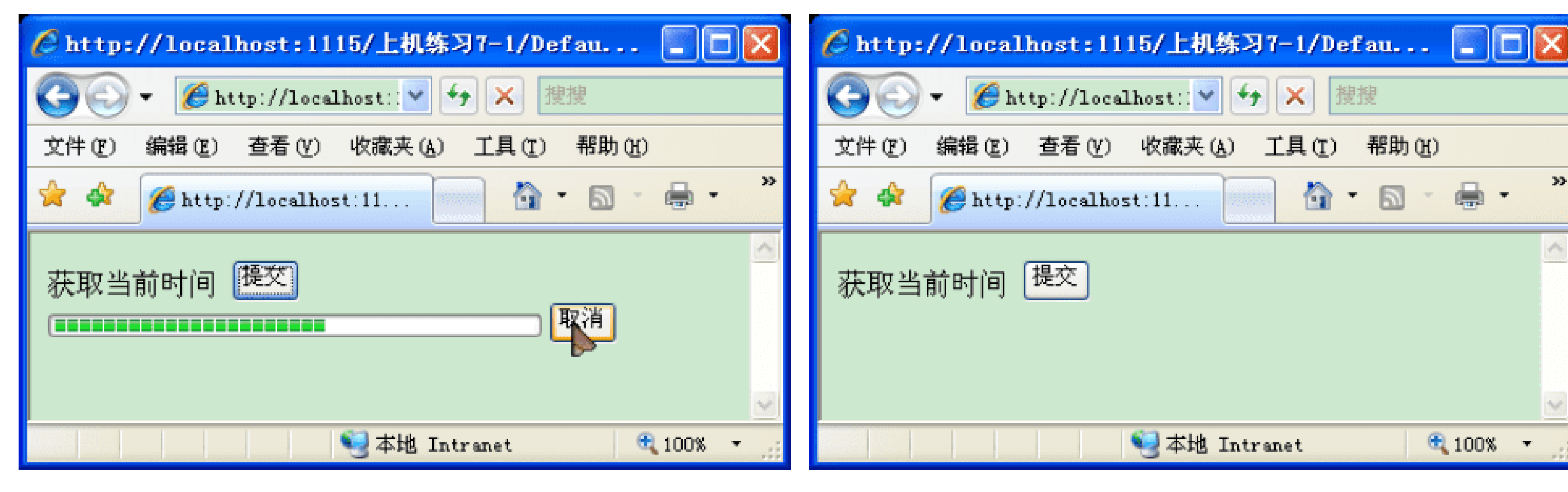


图 7-10　取消异步回发



提示

请不要将客户端方法与服务器端的事件处理相混淆：客户端方法允许浏览器捕获相应的事件，并使用 JavaScript 代码进行处理。这一过程根本不涉及到服务器端。事实上，当用户取消一个操作时，服务器端仍然或继续处理该请求，只是浏览器此时已关闭连接并停止监听。

7.4.2　定时更新多个 UpdatePanel

在下面的上机练习中，将使用一个 Timer 控件定时更新两个 UpdatePanel 控件，Timer 控件被放在 UpdatePanel 控件的外面，并被配置为 UpdatePanel 的触发器。

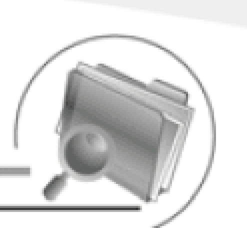
(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 7-2】。

(2) 在 Default.aspx 的设计视图中，添加一个 ScriptManager 控件和两个 UpdatePanel 控件，在 UpdatePanel 控件中添加一个 Label 控件，用于显示当前时间。

(3) 在两个 UpdatePanel 控件之外，添加一个 Timer 控件，设置其 Interval 属性为“1000”，即每隔 1 秒更新一次，并设置它为两个 UpdatePanel 控件的异步触发器。

(4) 此时源视图中生成的代码如下：

```
<div>
    <asp:ScriptManager ID="ScriptManager1" runat="server">
    </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
```



```
            <ContentTemplate>
                <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>
        <asp:UpdatePanel ID="UpdatePanel2" runat="server">
            <ContentTemplate>
                <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label>
            </ContentTemplate>
            <Triggers>
                <asp:AsyncPostBackTrigger ControlID="Timer1" EventName="Tick" />
            </Triggers>
        </asp:UpdatePanel>
        <asp:Timer ID="Timer1" runat="server" Interval="1000" ontick="Timer1_Tick">
        </asp:Timer>
    </div>
```

(5) 添加 Timer 控件的 Tick 事件处理程序，代码如下：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    Label1.Text = "UpdatePanel1  更新于：  " + DateTime.Now.ToString();
    Label2.Text = "UpdatePanel2  更新于：  " + DateTime.Now.ToString();
}
```

(6) 编译并运行程序，可以看到每间隔 1 秒同时刷新两个 UpdatePanel 控件，如图 7-11 所示。

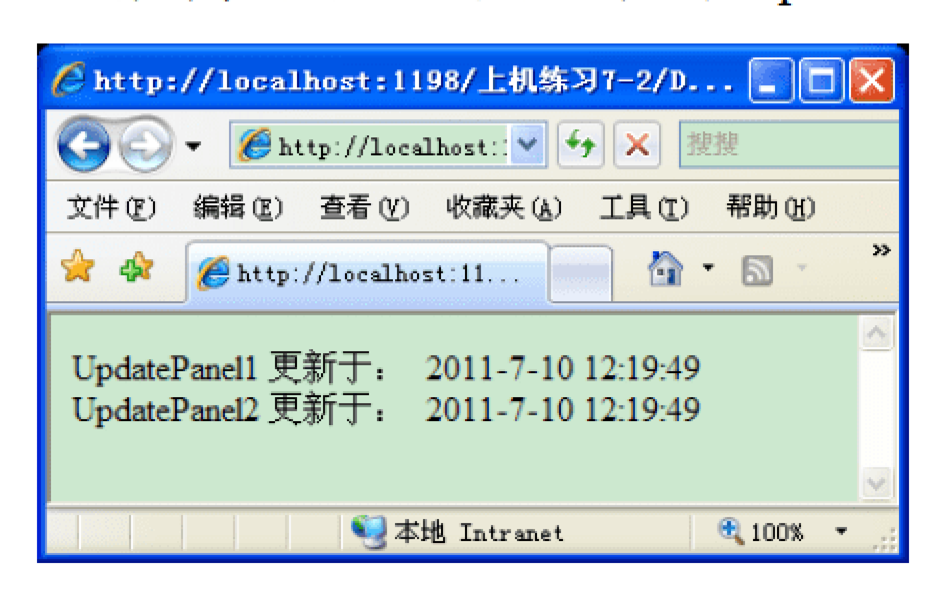


图 7-11　同时刷新两个 UpdatePanel 控件



7.5　习题

1. 工具箱中的 AJAX Extensions 类别中有 ScriptManager 和 ScriptManagerProxy 控件。解释这两个控件之间的区别，并解释何时要使用 ScriptManager，何时使用 ScriptManagerProxy。

2. UpdatePanel 控件有什么作用？如何让 UpdatePanel 控件外部的按钮进行异步刷新？

3. 如何让用户知道部分页面更新正在进行？

4. UpdateProgress 控件的 AssociatedUpdatePanelID 属性有什么用？

5. $addHandler 有什么作用？

6. 上机操作：

- 新建一个网站，在 Default.aspx 页面中的左上角显示当前的时间，要求采用局部刷新技术。
- 添加一个网页，实现相册功能，通过“上一个”和“下一个”按钮，局部刷新 Image 控件，显示不同的图片。
- 添加一个网页，实现局部刷新，要求刷新过程中给用户提供反馈。
- 在上面的进度条控件中，添加支持“取消”命令按钮的功能。



XML 和 Web 服务

学习目标

XML(eXtensible Markup Language，扩展标记语言)，.NET 把 XML 作为应用程序之间传递数据的一种主要方法。本章将介绍 XML 的基本概念和如何访问 XML。Web 服务奠定了下一代 Web 应用程序的基础，接下来本章用大量的篇幅详细介绍了 Web 服务的基本概念以及如何创建和调用 Web 服务，包括在 AJAX 站点中使用 Web 服务。

本章重点

- 了解 XML 的基本概念
- 掌握如何利用 ADO.NET 访问 XML
- Web 服务的工作原理
- 创建 Web 服务
- 调用 Web 服务
- 支持 AJAX 的 Web 服务

8.1 XML 概述

XML 是一种允许用户用来创建自己的标记的标记语言。它由万维网协会(W3C)创建，用来克服 HTML 的局限。和 HTML 一样，XML 基于 SGML——标准通用标记语言(Standard Generalized Markup Language)。XML 是 SGML 上的一个子集，XML 包含了 SGML 的很多特性，但是比 SGML 简单得多。

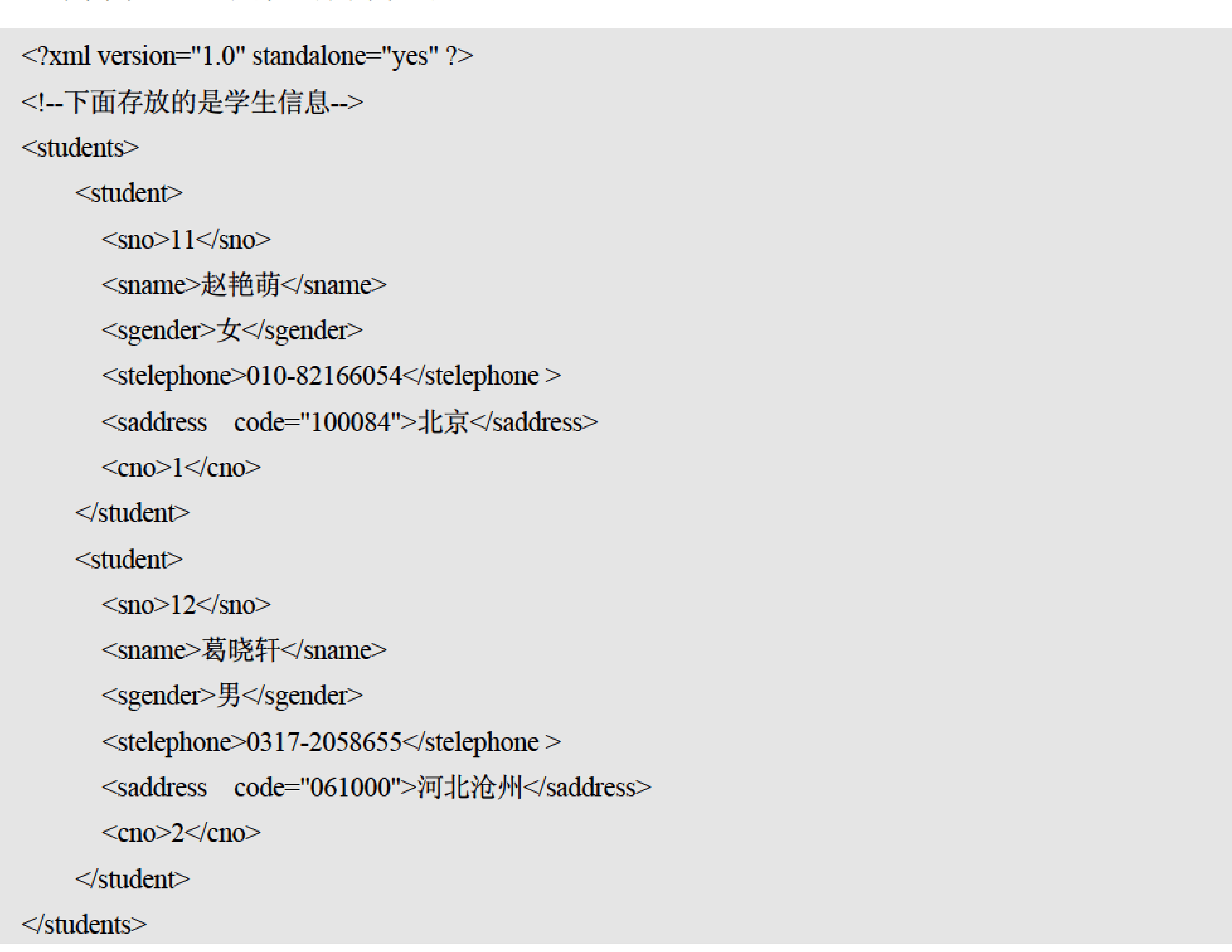
XML 是一种类似于 HTML 的标记语言，但是 XML 不是 HTML 的替代品，XML 和 HTML 是两种不同用途的语言，其中最主要的区别是：XML 是专门用来描述文本的结构，而不是用于描述如何显示文本的，而 HTML 则是用来描述如何显示文本的。

XML 提供了一种保存数据的格式，数据可以通过这种格式很容易地在不同的应用程序之间实现共享。它是用来存放数据的，可以利用相关的 XML API(MSXML DOM、JAVA DOM 等)对 XML 进行存取和查询。

8.1.1 XML 的基本结构

XML 不像 HTML 那样提供了一组事先已经定义好的标记，而是提供了一个标准。利用这个标准，用户可以根据需要定义自己的新标记。准确地说，XML 是一个元标记语言，它允许用户根据 XML 的规则，定义各种各样的标记语言。

一个简单 XML 文档的例子如下：

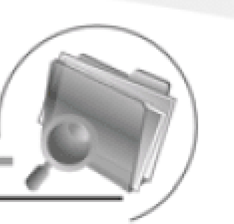


```
<?xml version="1.0" standalone="yes" ?>
<!--下面存放的是学生信息-->
<students>
    <student>
        <sno>11</sno>
        <sname>赵艳萌</sname>
        <sgender>女</sgender>
        <stelephone>010-82166054</stelephone >
        <saddress   code="100084">北京</saddress>
        <cno>1</cno>
    </student>
    <student>
        <sno>12</sno>
        <sname>葛晓轩</sname>
        <sgender>男</sgender>
        <stelephone>0317-2058655</stelephone >
        <saddress   code="061000">河北沧州</saddress>
        <cno>2</cno>
    </student>
</students>
```

XML 文档中的有效对象包括元素、根元素、子元素、处理指令、注释和属性。

⊙ 元素、根元素和子元素

元素是 XML 文档内容的基本单元。元素由起始标签、元素内容和结束标签组成。用户把要描述的数据对象放在起始标签和结束标签之间，如果一个元素从文件头的序言部分之后开始一直到文件结尾，包含了文件中所有的数据信息，那么就称之为根元素，被嵌套在其内的元素就称为子元素。如上面的 XML 文档中，<students> 是根元素，它的子元素是<student>，<student>又包含 6 个子元素：<sno>、<name>、< sgender>、<stelephonebirth >、<saddress>和<cno>。



◉ 处理指令

处理指令给 XML 解析器提供信息，使其能够正确解释文档内容，它的起始标识是“<?”，结束标识是“?>”。常见的 XML 声明就是一个处理指令：<?xml version = "1.0"?>。

处理指令还可以有其他用途，如定义文档的编码方式是 GB 码还是 Unicode 编码方式等。

◉ 注释

注释是 XML 文件中用作解释的字符数据，XML 处理器不对它们进行任何处理。注释是用“<!--”和“ --> ”括起来的，可以出现在 XML 元素间的任何地方，但是不可以嵌套，如上例中的“<!--下面存放的是学生信息--> ”就是注释。

◉ 属性

属性给元素提供进一步的说明信息，它必须出现在起始标签中。属性以名称/取值对出现，属性名不能重复，名称与取值之间用等号“=”分隔，并用引号把取值引起来。在上面的示例中，code 就是<saddress>元素的属性。

为了使一个 XML 文档结构完整，XML 必须遵守一定的规则。常见的 XML 文档规则如下：

(1) 文档必须以 XML 版本声明开始。

(2) 含有数据的元素必须有起始标记和结束标记。每个起始标记必须以相应的结束标记结束。如果一个文档未能结束一个标记，浏览器将报错。

(3) 不含数据并且仅使用一个标记的元素必须以“/>”结束。

(4) 文档只能有一个根元素。如上例中的<students>元素。

(5) 元素只能嵌套不能重叠。

(6) 属性值必须加引号。如：<saddress code="100084">。



8.1.2　XML 应用与发展前景

XML 自推出以来，尤其是在 1998 年 2 月成为 W3C 推荐标准以来，受到了广泛的支持。各大软件厂商如 IBM、Microsoft、Oracle 等都积极支持并参与 XML 的研究和产品化工作，先后推出了支持 XML 的产品或者改造原有的产品以支持 XML，W3C 也一直致力于完善 XML 的整个理论体系。

XML 虽然获得了极大的支持，但是它还有一段路要走。首先，XML 的规则还有许多技术细节没有解决。其次，现在虽然出现了一些 XML 工具和应用，但是其市场反应还有待进一步观察。另外，XML 的出现和迅猛发展并不意味着 HTML 即将退出互联网舞台。由于 HTML 的易学易用和具有非常多的工具支持，HTML 将在较长的时间里继续在 Web 舞台上充当主角。但如果用户想超越 HTML 的范围，XML 将是最佳的选择。

无论如何，XML 的出现使互联网跨入了一个新的阶段，它已成为因特网领域中一个重要的数据交换标准和开发平台。没有 XML 就没有 Web 服务，也就没有今天构建应用软件的轰轰烈烈的 SOA(Service Oriented Architecture)。XML 的诞生已经而且将继续促使全新种类的基础架构和应用程序的产生，而这些新的基础架构和应用程序又将需要新的软件和硬件工具。可以预测，无

论是在软件还是硬件上，XML 都将开辟一系列的新市场，促成互联网上新的革命。

作为互联网的新技术，XML 的应用非常广泛，可以说 XML 已经渗透到了互联网的各个角落。考察现在的 XML 应用，可以分为以下几个方面。

1. 数据交换

利用 XML 在应用程序之间作数据交换已不是什么秘密了，毫无疑问应被列为第一位。那么为什么 XML 在这个领域里的地位这么重要呢？原因就是 XML 使用元素和属性来描述数据。在数据传送过程中，XML 始终保留了诸如父、子关系这样的数据结构。几个应用程序可以共享和解析同一个 XML 文件，不必使用传统的字符串解析或拆解过程。

相反，普通文件不对每个数据段做描述，也不保留数据关系结构。使用 XML 做数据交换可以使应用程序更具有弹性，因为可以用位置(与普通文件一样)或用元素(从数据库)来存取 XML 数据。

2. Web 服务

Web 服务是最令人激动的革命之一，它让使用不同系统和不同编程语言的人们能够相互交流和分享数据。其基础在于 Web 服务器用 XML 在系统之间交换数据。

Web 服务使用到了 SOAP 协议。SOAP 协议是一套用于 Web 服务端和客户端通信的标准消息控制协议，它用 XML 构造消息，消息中包含了服务端和客户端所需要的参数或值。

SOAP 协议可以在用不同编程语言构造的对象之间传递消息。这意味着一个 C#对象能够与一个 Java 对象进行通信。这种通信甚至可以发生在运行于不同操作系统上的对象之间。DCOM、CORBA 或 Java RMI 只能在紧密耦合的对象之间传递消息，SOAP 协议则可在松耦合对象之间传递消息。

3. 内容管理

XML 只用元素和属性来描述数据，而不提供数据的显示方法。这样，XML 就提供了一个优秀的方法来标记独立于平台和语言的内容。

使用像 XSLT 这样的语言能够轻易地将 XML 文件转换成各种格式文件，比如 HTML、WML、PDF、EDI 等。XML 具有的能够运行于不同系统平台之间和转换成不同格式目标文件的能力使得它成为内容管理应用系统中的优秀选择。

4. Web 集成

现在有越来越多的设备支持 XML 了，这使得 Web 开发商可以在个人电子助理和浏览器之间用 XML 来传递数据。将 XML 文本直接送进这样的设备，可以使用户能够自己掌握数据显示方式。常规的客户/服务(C/S)方式为了获得数据排序或更换显示格式，必须向服务器发出申请，而 XML 则可以直接处理数据，不必经过向服务器申请查询、返回结果这样的双向“旅程”，同时设备也不需要配制数据库，甚至还可以对设备上的 XML 文件进行修改并将结果返回给服务器。



5. 配置文件

许多应用都将配置数据存储在各种文件中，比如.INI 文件。虽然这样的文件格式已经使用多年并一直很好用，但是 XML 还是以更为优秀的方式为应用程序标记配制数据。使用.NET 中的类，如 XmlDocument 和 XmlTextReader，将配置数据标记为 XML 格式，能使其更具可读性，并能方便地集成到应用系统中去。使用 XML 配置文件的应用程序能够方便地处理所需数据，不用像其他应用那样要经过重新编译才能修改和维护应用系统。

8.2　使用 ADO.NET 访问 XML

在前面已经讨论了如何使用 ADO.NET 访问数据库的问题。数据库是进行数据存储和管理的一种习惯的方式，现在，XML 已逐步成为数据存储的一种新的方式，因此可以考虑将数据保存在 XML 文档中，并采用一定的方法对它进行管理。ADO.NET 提供了对 XML 数据访问的功能。下面就介绍如何使用 ADO.NET 访问 XML 数据。



8.2.1　读写 XML 文件

使用 DataSet 的 ReadXml 方法可以读取 XML 文档的所有数据，使用 WriteXml 方法可以将数据保存到 XML 文件中。下面通过一个例子来说明读写 XML 文件的方法。

【例 8-1】读写 XML 文档。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 8-1】。

(2) 在【解决方案资源管理器】面板中，右击【例 8-1】解决方案，从弹出的快捷菜单中选择【添加新项】命令，添加一个名为 Students.xml 的 XML 文件，文件的内容如前面的 XML 文件示例所示，删除<saddress>元素的 code 属性。

(3) 打开 Default.aspx 文件的设计视图，添加两个 Button 控件和一个 GridView 控件，Button 控件的 Text 属性分别为“读取数据”和“修改并保存数据”。

(4) 为两个按钮添加单击事件处理程序，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    DataSet ds = new DataSet();
    ds.ReadXml(Server.MapPath("Students.xml"));//读取 XML 数据到 DataSet 数据集
    GridView1.DataSource = ds.Tables[0].DefaultView;//绑定数据源
    GridView1.DataBind();
}
protected void Button2_Click(object sender, EventArgs e)
{
```

```
        DataSet ds = new DataSet();
        ds.ReadXml(Server.MapPath("Students.xml"));
        //将数据表 ds 的数据复制到 DataTable 对象
        DataTable dtable = ds.Tables[0];
        //用 DataRowCollection 对象获取这个数据表的所有数据行
        DataRowCollection rows = dtable.Rows;
        //修改操作，逐行遍历，取出各行的数据
        for (int i = 0; i < rows.Count; i++)
        {
            DataRow row = rows[i];
            //给每位学生的学号加 1
            row[0] = Int32.Parse(row[0].ToString()) + 1;
        }
        ds.WriteXml(Server.MapPath("Students.xml"));//将 DataSet 数据写成 XML 文本
        //绑定数据源，重新加载并显示
        GridView1.DataSource = ds.Tables[0].DefaultView;
        GridView1.DataBind();
    }
```

(5) 编译并运行程序，启动默认浏览器打开 Default.aspx 文件，单击【读取数据】按钮，读取 XML 文件中的内容显示到 GridView 控件中，如图 8-1 所示。

(6) 单击【修改并保存数据】按钮，将把每个学生的学号都加 1 并保存，然后重新加载并显示，如图 8-2 所示。

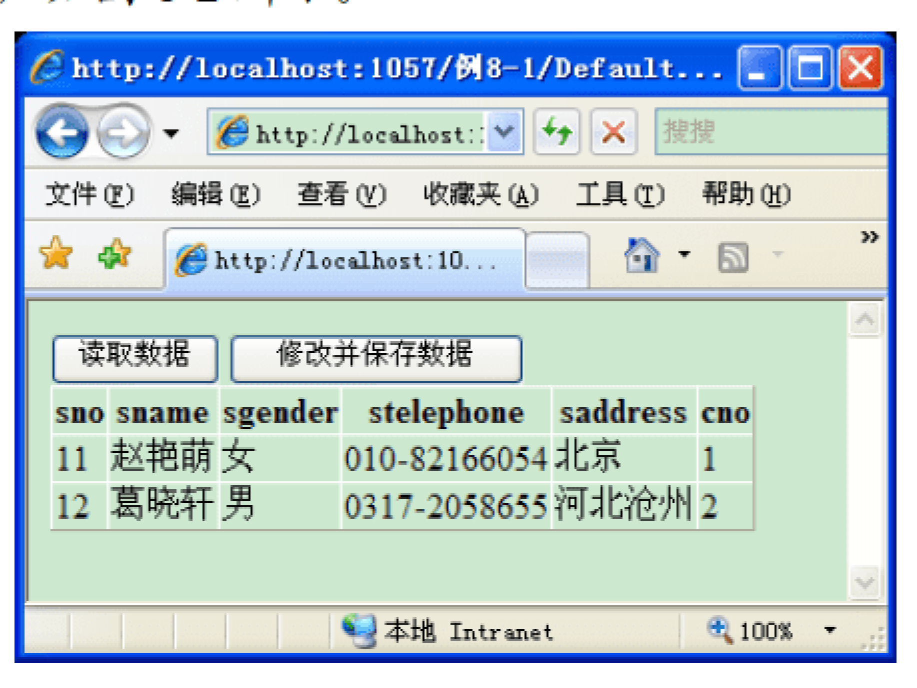


图 8-1　读取 XML 文件中的数据

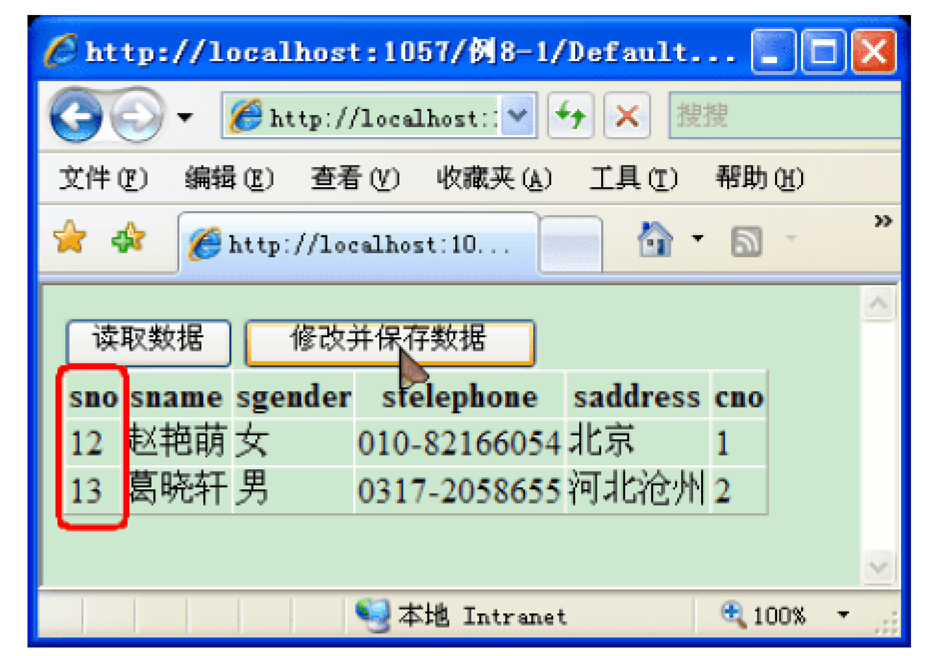


图 8-2　修改并保存数据



知识点

通过例 8-1 可以看到，使用 ADO.NET 编辑 XML 文档，实际上就是对 DataSet 数据集数据的编辑。



8.2.2　将数据库数据转换成 XML

如果数据集 DataSet 中填充的是数据库中的数据，那么调用 WriteXml 方法后，即可将数据库数据转换成 XML 文档，例 8-2 演示了这一操作。

【例 8-2】将数据库数据转换成 XML 文档。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 8-2】。

(2) 切换到 Default.aspx 文件的设计视图，添加一个 Button 控件、一个 Label 控件和一个 GridView 控件，设置 Button 控件的 Text 属性为“保存数据到 XML”。

(3) 切换到后台代码文件，添加 ADO.NET 命空间的引用，代码如下：

```
using System.Data.SqlClient;
using System.Data;
```



(4) 为按钮添加单击事件处理程序，使用 SqlDataAdapter 填充数据集，将学生表 Student 中的数据保存到 XML 文件中，并显示在 GridView 控件中，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    string strConnect = "Data Source=zhao\\tsinghua;Initial Catalog=InfoManage;Integrated
Security=True;user=sa;password=Sapassword";
    SqlConnection con = new SqlConnection(strConnect);
    DataSet ds = new DataSet();
    try
    {
        con.Open();
        SqlDataAdapter sqld = new SqlDataAdapter("select * from student", con);
        sqld.Fill(ds, "student");//用 Fill 方法填充 DataSet
        sqld.Update(ds, "student");
        GridView1.DataSource = ds.Tables["student"];
        GridView1.DataBind();
        ds.WriteXml(Server.MapPath("stuinfo.xml"));
        Label1.Text = "保存 XML 文件成功！";
    }
    catch (Exception ex)
    {
        Label1.Text = "操作失败，失败原因：" + ex.Message;
    }
    finally
    {
        con.Close();
```

```
            con = null;
        }
    }
```

(5) 编译并运行程序，在浏览器中加载 Default.aspx 页面，单击【保存数据到 XML】按钮，效果如图 8-3 所示。

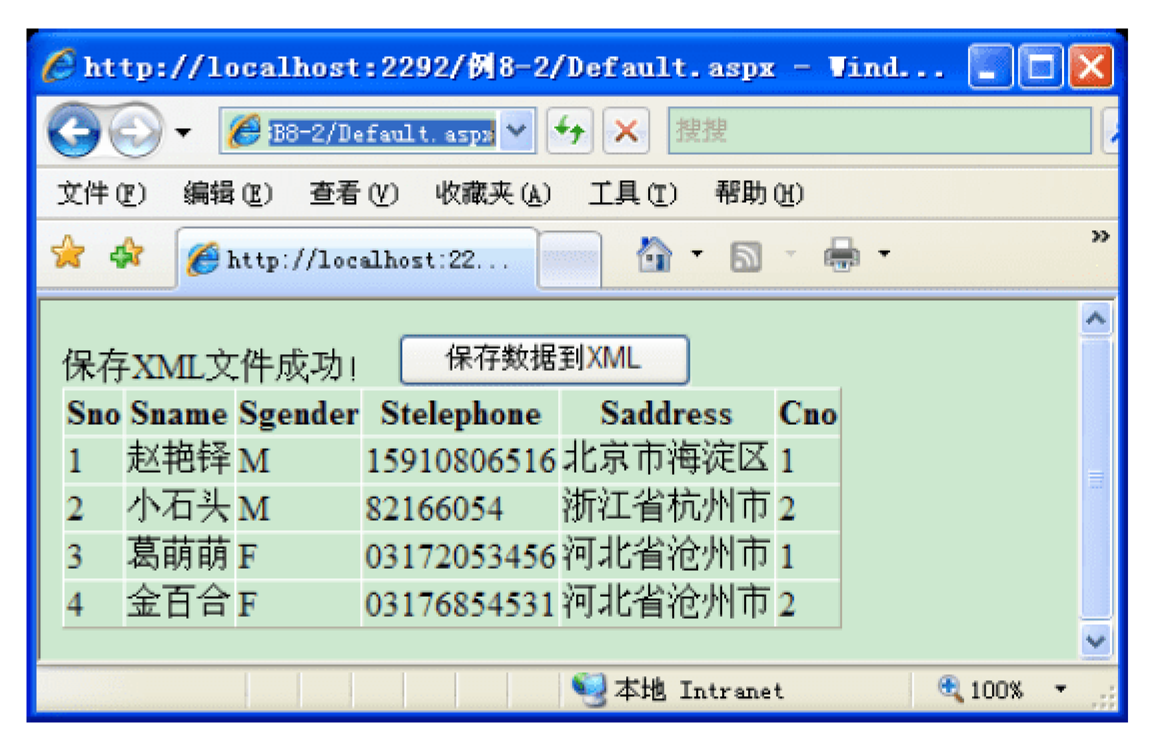
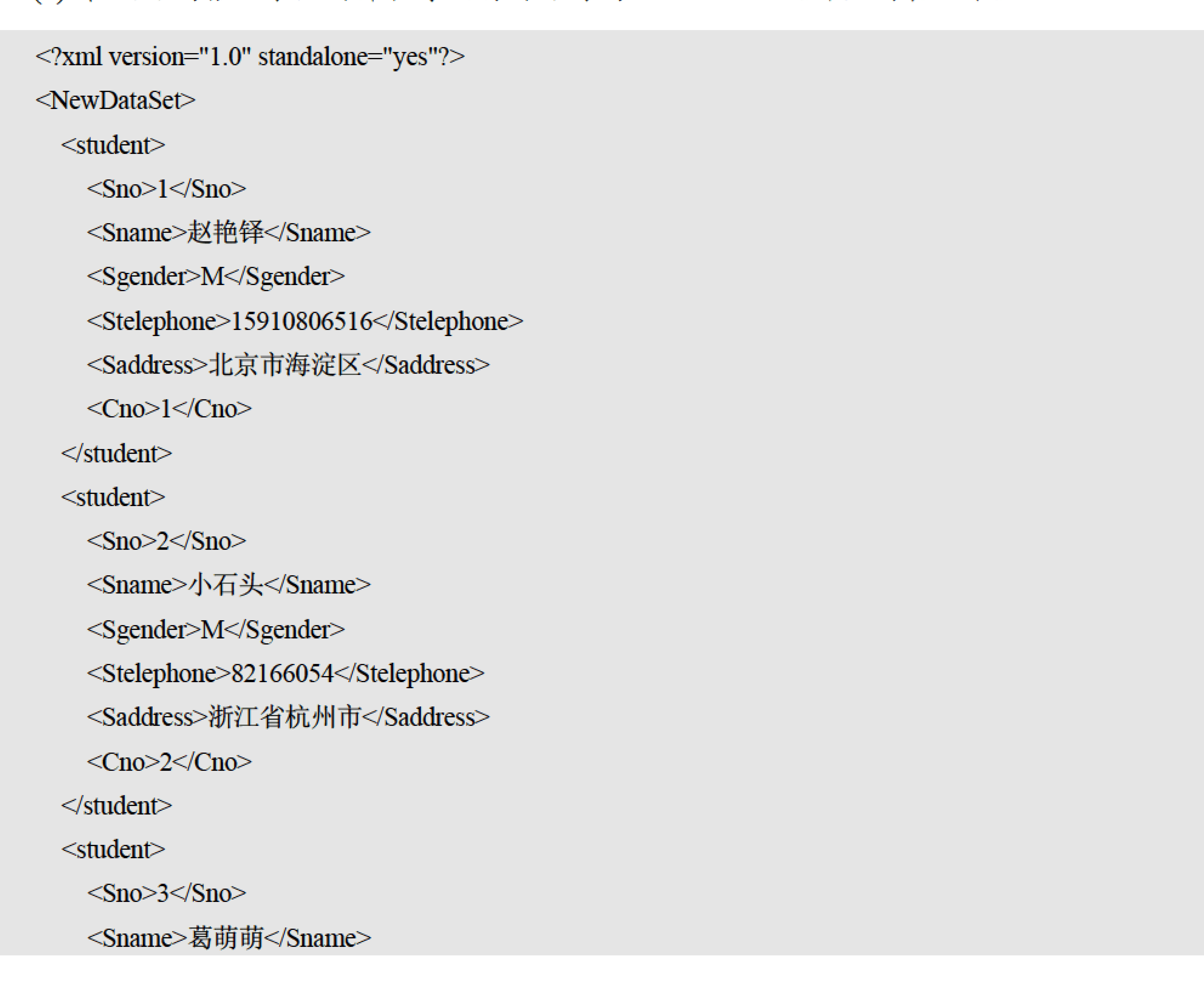


图 8-3　页面运行效果

(6) 在网站的根目录，打开程序运行时保存的 stuinfo.xml 文件，内容如下：



```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <student>
    <Sno>1</Sno>
    <Sname>赵艳铎</Sname>
    <Sgender>M</Sgender>
    <Stelephone>15910806516</Stelephone>
    <Saddress>北京市海淀区</Saddress>
    <Cno>1</Cno>
  </student>
  <student>
    <Sno>2</Sno>
    <Sname>小石头</Sname>
    <Sgender>M</Sgender>
    <Stelephone>82166054</Stelephone>
    <Saddress>浙江省杭州市</Saddress>
    <Cno>2</Cno>
  </student>
  <student>
    <Sno>3</Sno>
    <Sname>葛萌萌</Sname>
```



```
        <Sgender>F</Sgender>
        <Stelephone>03172053456</Stelephone>
        <Saddress>河北省沧州市</Saddress>
        <Cno>1</Cno>
    </student>
    <student>
        <Sno>4</Sno>
        <Sname>金百合</Sname>
        <Sgender>F</Sgender>
        <Stelephone>03176854531</Stelephone>
        <Saddress>河北省沧州市</Saddress>
        <Cno>2</Cno>
    </student>
</NewDataSet>
```



可以看出，这个文档保存了 student 数据表中所有的数据。其中使用<NewDataSet>作为根结点标记，<student>作为每个记录的标记(student 是 sqld.Fill(ds, "student")语句中使用的名字)，另外，每个字段的名字作为数据元素的标记名。

8.3　Web 服务概述

Web 服务奠定了下一代 Web 应用程序的基础。客户应用程序无论是 Windows 应用程序，还是 ASP.NET Web 应用程序，无论客户程序是运行在 Windows、Pocket Windows 还是其他操作系统上，它们都可以通过 Internet 使用 Web 服务定期通信。Web 服务是服务器端的程序，用来监听来自客户应用程序的消息，并返回特定的信息。这些信息可能来自 Web 服务本身，也可能是同一个域中的其他组件，或其他 Web 服务。

8.3.1　什么是 Web 服务

简单地讲，Web 服务是一个基于因特网的可通过 Web 被远程调用的应用程序模块(API)，例如若网站想提供天气预报服务，并不用自己实现天气预报功能，只需调用其他公司提供的免费或付费 Web 服务即可。

- 服务就是一个软件，它和客户端应用程序没有很紧密地耦合或关联。服务是可以被动态地发现及组合成其他软件的软件实体。
- Web 服务是一种基于 XML、JSON、SOAP、HTTP、UDDI、WSDL 等一系列标准实现的分布式计算技术和软件组件。

- Web 服务提供了一个松耦合和跨平台的分布式计算环境，它是一个与操作系统无关、程序设计语言无关、机器类型无关、运行环境无关的平台，实现网络上应用的共享，并可用于复杂的系统集成。

微软为 Web 服务下的定义是通过标准的 Web 协议可编程访问的 Web 组件。每个 Web 服务的实现是完全独立的。Web 服务具有基于组件的开发和 Web 开发两者的优点，是 Microsoft 的.NET 程序设计模式的核心。

国际标准化组织 W3C 为 Web 服务下的定义是一个通过 URL 识别的软件应用程序，其界面及绑定能用 XML 文档来定义、描述和发现，使用基于 Internet 协议上的消息传递方式与其他应用程序进行直接交互。

1. Web 服务的影响

(1) Web 服务支持在 Web 站点上放置可编程的元素，用户可以抓取已有的元素，构成自己的新服务。

(2) 能进行基于 Web 的分布式计算和处理，能很好地兼容现有的 Web 技术。

(3) Web 服务使得 Internet 成为一个可以无限扩展、拥有无限潜力的分布式计算平台。

(4) 任何设备可以随时随地访问 Internet 上的 Web 服务。

(5) 软件模块充分复用、计算机资源充分共享、信息无缝共享和交流。

(6) 利用 Web 服务，公司和个人将能够迅速且廉价地向整个国际互联网络提供他们的服务，进而建立全球范围的联系，在广泛的范围内寻找可能的合作伙伴。

2. Web 服务的主要特征

(1) 互操作性：一个 Web 服务可与其他 Web 服务交互，协同工作；可以使用任何语言开发 Web 服务或使用他人提供的 Web 服务；开发环境可以异构。

(2) 普遍性：Web 服务使用 HTTP 和 XML 进行通信，支持这些技术的设备都可以拥有和访问 Web 服务。

(3) 松散耦合：Web 服务的实现对使用者透明，当服务的实现发生变动时不影响用户使用。

(4) 高度可集成能力：Web 服务和采用了简单的、易理解的标准 Web 协议作为组件界面描述和协同描述规范，屏蔽了平台的异构性，CORBA、DCOM、EJB 等都可通过它进行互操作。

8.3.2 ASP.NET Web 服务体系

.NET 平台和 ASP.NET 在创建和使用 Web 服务方面提供了广泛的支持。这些技术赋予用户一个优秀的、简单易用的平台，从而可以快速有效地创建和使用 Web 服务。如图 8-4 所示就是 ASP.NET Web 服务的体系结构。

ASP.NET Web 服务体系包括客户端应用程序、ASP.NET Web 服务程序以及一些文件，如：代码文件、.asmx 文件和编译后的.dll 文件；还包括一台 Web 服务器来存载 Web 服务程序和客户



端。如果需要，还可以有一台数据库服务器来存取 Web 服务中的数据。

SOA(Service Oriented Architecture)Web 服务的架构如图 8-5 所示。其中包括服务提供者、服务请求者、服务中介者 3 个参与者和发布、发现、绑定 3 个基本操作。

XML 或 JSON 是数据的格式，SOAP 是调用 Web 服务的协议，WSDL 是描述 Web 服务的格式，而 UDDI 是 Web 服务发布、查找和利用的组合。

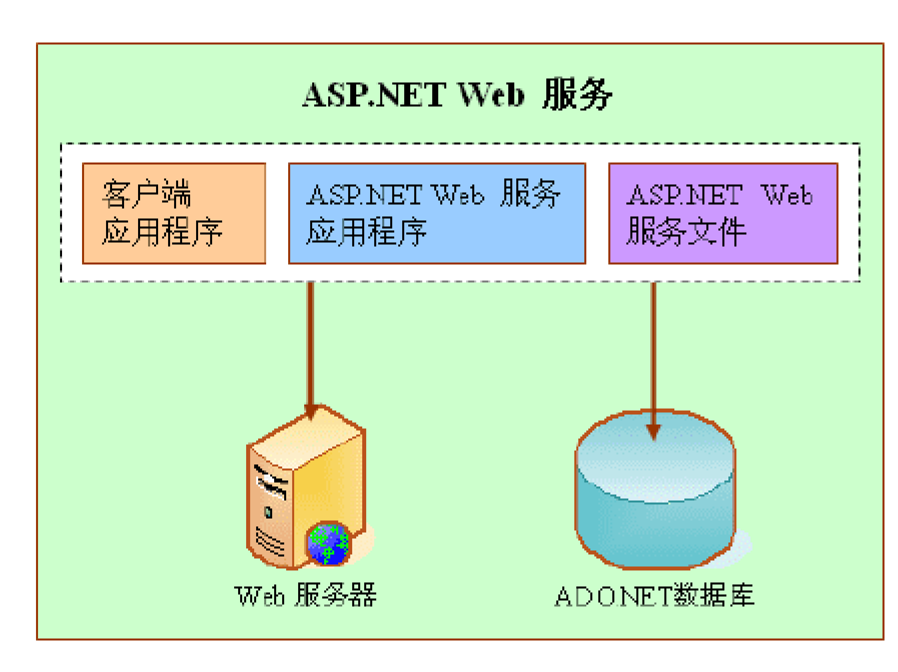


图 8-4 ASP.NET Web 服务体系

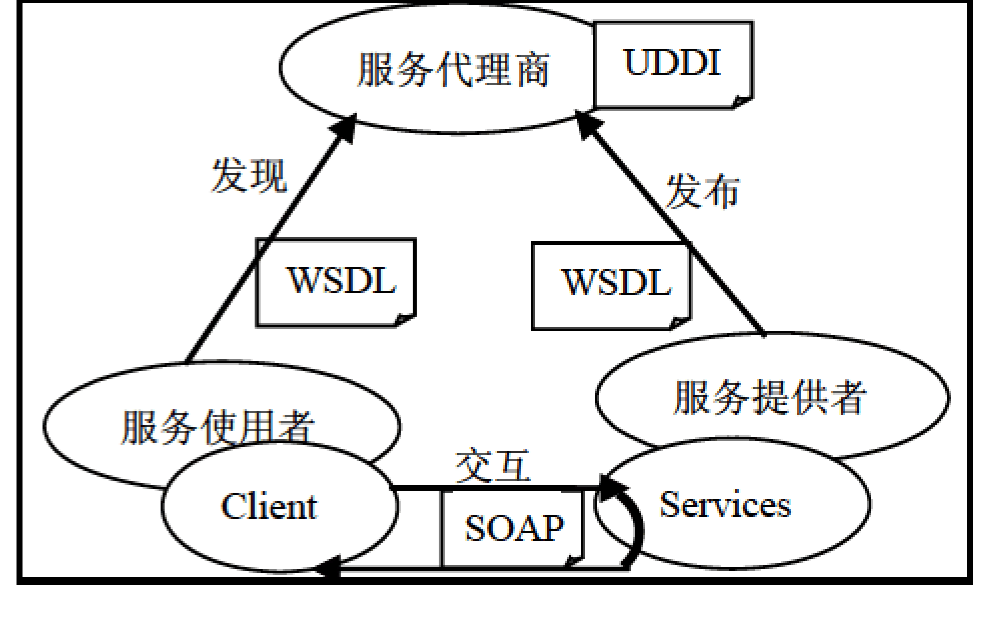


图 8-5 SOA Web 服务架构



- SOAP(Simple Object Access Protocol)：SOAP 是一套用于 Web 服务端和客户端通信的标准消息控制协议，SOAP 用 XML 构造消息，消息中包含了服务端和客户端所需要的参数或值。
- WSDL(Web Services Description Language)：WSDL 是 Web 服务描述语言。可以认为 WSDL 文件是一个 XML 文档，用于说明一组 SOAP 消息以及如何交换这些消息。换句话说，WSDL 对于 SOAP 的作用就像 IDL 对于 CORBA 或 COM 的作用。通常 WSDL 文档由软件生成和使用。
- UDDI(Universal Description Discovery and Integration)：UDDI 是 Web 服务的黄页。与传统黄页一样，用户可以搜索提供所需服务的公司，阅读以了解所提供的服务，然后与某人联系以获得更多信息。当然用户也可以提供 Web 服务而不在 UDDI 中注册，就像在地下室开展业务一样，依靠的是口头吆喝；但是如果希望拓展市场，则需要 UDDI 以便能被客户发现。

8.3.3 支持 AJAX 的 Web 服务

ASP.NET AJAX 提供了完整的架构以从客户端 JavaScript 调用 ASP.NET Web 服务。设计者可以轻松地用 AJAX 把服务器端的数据和功能集成到用户响应的 Web 页面中，而所需要做的就是仅仅用[ScriptService]属性来标识 Web 服务。ASP.NET AJAX 框架会为 Web 服务自动生成 JavaScript 代理，然后通过使用代理来调用 Web 方法。

JSON(JavaScript Object Notation)是一种轻量级的数据交换格式，易于阅读和编写，易于机器解析和生成。JSON 采用完全独立于语言的文本格式，使用了类似于 C 语言家族的习惯。这些特

性使 JSON 成为理想的数据交换语言。

JSON 建构于以下两种结构：

(1) “名称/值”对的集合。不同的语言中，它被理解为对象(object)、记录(record)、结构(struct)、字典(dictionary)、哈希表(hash table)、有键列表(keyed list)或者关联数组(associative array)。

(2) 值的有序列表。在大部分语言中，它被理解为数组(array)。

这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得一种数据格式在同样基于这些结构的编程语言之间交换成为可能。

JSON 以一种特定的字符串形式来表示 JavaScript 对象。如果将具有这样一种形式的字符串赋给任意一个 JavaScript 变量，那么该变量会变成一个对象引用，而这个对象就是字符串所构建出来的。

例如：假设需要创建一个 User 对象，并具有用户 ID、用户名、用户 Email 这 3 个属性，可以使用以下 JSON 形式来表示 User 对象：


```
{"UserID":110, "Name":"葛萌萌", "Email":gemm@gmail.com"};
```

如果把这一字符串赋予一个 JavaScript 变量，那么就可以直接使用对象的任一属性了。完整代码如下：

```
<script>var User = {"UserID":110, "Name":"Sunny", "Email":":happy@gmail.com"};
alert(User.Name); </script>
```

借助 ASP.NET AJAX Extension，微软选择 JSON 在服务器和 AJAX 客户端实现数据交换，从而创建支持 AJAX 的 Web 服务。在客户端和服务器端都实现了数据的串行化器和并行化器以使数据按 JSON 的格式交换。网页中的客户端脚本与服务器通过 Web 服务通信层进行通信来访问 Web 服务，该通信层使用 AJAX 技术进行 Web 服务调用，数据在客户端与服务器之间通常采用 JSON 格式进行异步交换。



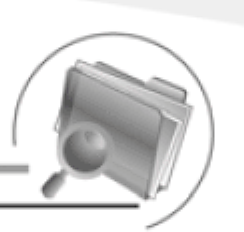
知识点

默认将 JavaScript 对象序列化为 JSON 格式，使用 JavaScript 的 eval 函数可以进行反序列化。但 Web 服务和 ASP.NET 网页中的单个方法可以返回其他格式。可以通过 ScriptMethod 属性来指定方法的序列化格式。对于某个 ASMX 服务，可以设置 ScriptMethod 属性形如[ScriptMethod(ResponseFormat = ResponseFormat.Xml)]，以使某个 Web 服务方法返回 XML 数据。



8.4 创建和调用 Web 服务

在.NET Framework 中，可以很容易创建和使用 Web 服务。与 Web 服务相关的命名空间有以下 3 个。



- System.Web.Services：该命名空间中的类用于创建 Web 服务。
- System.Web.Services.Description：使用该命名空间可以通过 WSDL 描述 Web 服务。
- System.Web.Services.Protocols：使用该命名空间可以创建 SOAP 请求和响应。

8.4.1　WebService 类

要创建 Web 服务，可以从 System.Web.Services.WebService 中派生 Web 服务类。WebService 类提供了对 ASP.NET Application 和 Session 对象的访问。WebService 类的常用属性如表 8-1 所示。

表 8-1　WebService 类的常用属性

属　性	说　明
Application	为当前请求返回一个 HttpApplicationState 对象
Context	返回一个封装 HTTP 特定信息的 HttpContext 对象。可以从中读取 HTTP 标题信息
Server	返回一个 HttpServerUtility 对象。这个类有一些方法，可以进行 URL 编码和解码
Session	返回一个 HttpSessionState 对象，以存储客户端的一些状态
User	返回一个实现 IPrincipal 接口的用户对象。使用这个接口可以得到用户名和身份验证类型
SoapVersion	返回 Web 服务使用的 SOAP 版本。SOAP 版本封装在 SoapProtocolVersion 枚举中



1. WebService 属性

与普通的类继承不同的是，WebService 的子类需要用 WebService 属性来标记，该属性用于向 XML Web 服务添加附加信息，如描述其功能的字符串。这是一个 WebServiceAttribute 类的对象，共有 3 个可选属性，如表 8-2 所示。

表 8-2　WebServiceAttribute 类的属性

属　性	说　明
Description	服务的描述信息，可用于 WSDL 文档
Name	获取或设置 Web 服务名称
Namespace	获取或设置 Web 服务的 XML 命名空间。其默认值为 http://tempuri.org，用于测试，但在开发这个服务前，应修改该命名空间

2. WebServiceBinding 属性

.NET 2.0 给 Web 服务添加了一个新属性 WebServiceBinding。这个属性用于把 Web 服务标记为可交互操作的一致性级别。如表 8-3 所示列出了 WebServiceBindingAttribute 类的一些属性。

表 8-3　WebServiceBindingAttribute 类的属性

属　性	说　明
ConformanceClaims	Web 服务的一致性级别可以设置为 WsiClaims 枚举的一个值。WsiClaims 有两个值：Web 服务遵循 Basic Profile 1.0 时，其值为 BP10；没有定义任何一致性级别时，其值为 None
EmitConformanceClaims	EmitConformanceClaims 是一个布尔属性，定义了用 ConformanceClaims 属性指定的一致性级别是否应传送给生成的 WSDL 文档
Name	使用 Name 属性可以定义绑定的名称。该名称默认与 Web 服务相同，但要加上 Soap 字符串
Location	Location 属性定义了绑定消息的位置，例如 http://www.timeout.com/Webservice.asmx?wsdl
Namespace	Namespace 属性定义了绑定的 XML 命名空间

3. WebMethod 属性

Web 服务中可以使用的所有方法都必须用 WebMethod 属性来标记。当然，也可以有其他没有用 WebMethod 标记的方法，但这些方法只能在 WebMethod 中调用，而不能在客户机上调用。使用属性类 WebMethodAttribute，就可以在远程客户机上调用方法，并可以定义是否缓存响应，缓存时间有多长，会话状态是否与指定的参数一起存储等。WebMethod Attribute 类的属性如表 8-4 所示。

表 8-4　WebMethodAttribute 类的属性

属　性	说　明
BufferResponse	获取或设置响应是否应缓存的标志。默认值为 true。使用被缓存的响应，仅可以将已完成的软件包传递给客户机
CacheDuration	使用这个属性可以设置结果应缓存的时间长短。如果在这个属性设置的时间段中第二次发出了相同的请求，就返回缓存的值。默认值为 0，这表示结果不缓存
Description	该描述用于给预期的用户生成服务帮助页面
EnableSession	布尔值，表示会话状态是否有效。默认值是 false，因此 WebService 类的 Session 属性不能用于存储会话状态
MessageName	默认状态下，把消息名设置为方法名
TransactionOption	这个属性表示方法的事务处理支持。默认值是 Disabled

4. ScriptService 属性

System.Web.Script.Services.ScriptService 属性用于使用 ASP.NET AJAX 从脚本中调用 Web 服务。ScriptService 属性的主要参数如表 8-5 所示。

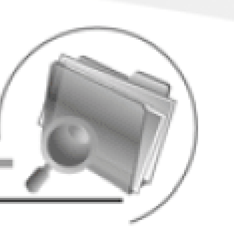


表 8-5　ScriptService 属性的主要参数

属　性	说　　明
ResponseFormat	指定是否将响应序列化为 JSON 或者 XML。默认为 JSON，但是，当方法的返回值是 XmlDocument 时，XML 格式会比较方便
UseHttpGet	表明是否可以使用 HTTP GET 调用 Web 服务方法。由于安全性原因，此项的默认设置为 false
XmlSerializeString	表明包括字符串在内的所有返回类型是否都序列化为 XML，默认为 false，当响应格式设置为 JSON 时，将忽略该属性的值

8.4.2　创建 Web 服务



使用 VWD 创建 Web 服务非常简单。与其他所有文档类型一样，VWD 也附带有 Web 服务模板。可以使用【添加新项】对话框来添加 Web 服务。然后可以修改服务，并在 Web 浏览器中使用 ASP.NET 运行库自动创建的标准测试页面测试它。当 Web 服务正确运行时，就可以调用该服务。

下面创建一个简单的 Web 服务，该 Web 服务提供了两个方法调用：HelloWord 和 AuthCode。其中 HelloWorld 是默认的，AuthCode 用于返回一个验证码图片的二进制数据。

【例 8-3】创建 Web 服务，返回字母和数字组成的验证码数据。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 8-3】。

(2) 在【解决方案资源管理器】面板中，右击【例 8-3】解决方案，从弹出的快捷菜单中选择【添加新项】命令，选择【Web 服务】模板，添加名为 WebService.asmx 的 Web 服务。

(3) 此时在网站的【App_Code】文件夹下会自动生成一个名为 WebService.cs 的文件，同时在网站目录中会生成 WebService.asmx 文件，如图 8-6 所示。

(4) 自动生成的 WebService.cs 文件的代码如下所示。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
/// <summary>
///WebService 的摘要说明
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
//若要允许使用 ASP.NET AJAX 从脚本中调用此 Web 服务，请取消对下行的注释。
```

```
// [System.Web.Script.Services.ScriptService]
public class WebService : System.Web.Services.WebService {
    public WebService () {
        //如果使用设计的组件，请取消注释以下行
        //InitializeComponent();
    }
    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }
}
```

从这段代码中可以找到前面介绍的 4 个属性：WebService、WebServiceBinding、ScriptService 和 WebMethod。其中，该 Web 服务提供了一个 HelloWorld 方法，调用该方法将返回字符串“Hello Word”。

(5) 无需添加和修改任何代码，即可启用和测试 Web 服务“HelloWorld”。编译并启动应用程序，在浏览器中打开 WebService.asmx 文件，显示服务支持的操作，如图 8-7 所示。

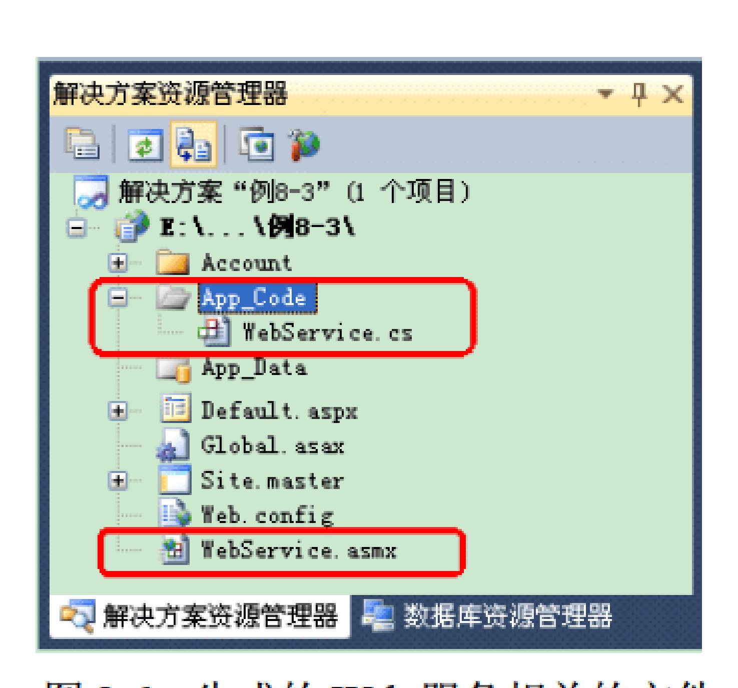


图 8-6　生成的 Web 服务相关的文件

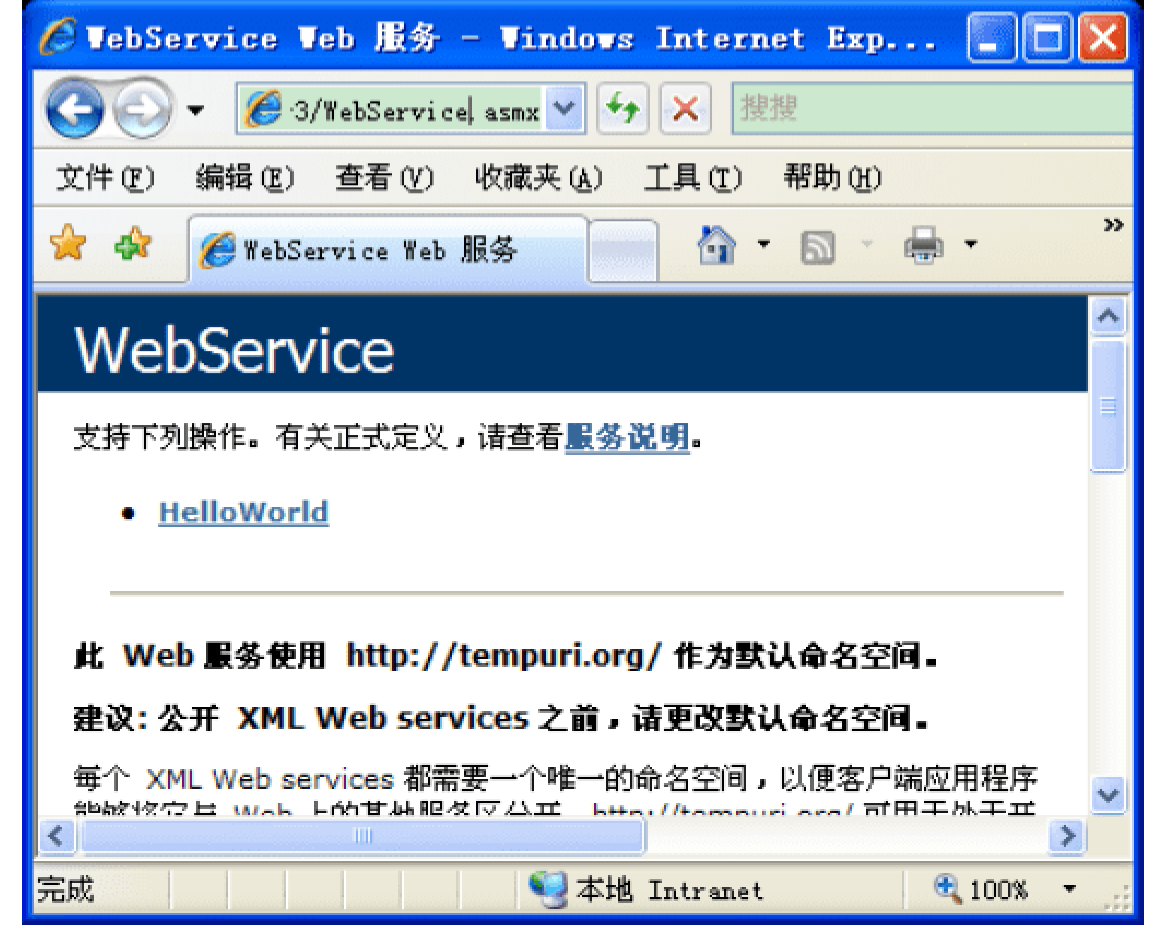
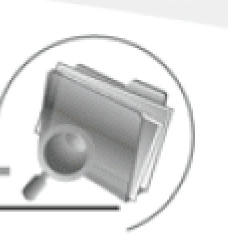


图 8-7　Web 服务的测试页面

(6) 单击“HelloWorld”链接将调用该方法，如图 8-8 所示。

(7) 因为该方法不需要任何参数，所以直接单击“调用”按钮即可返回调用结果，结果包含在一个 XML 文件中，如图 8-9 所示。



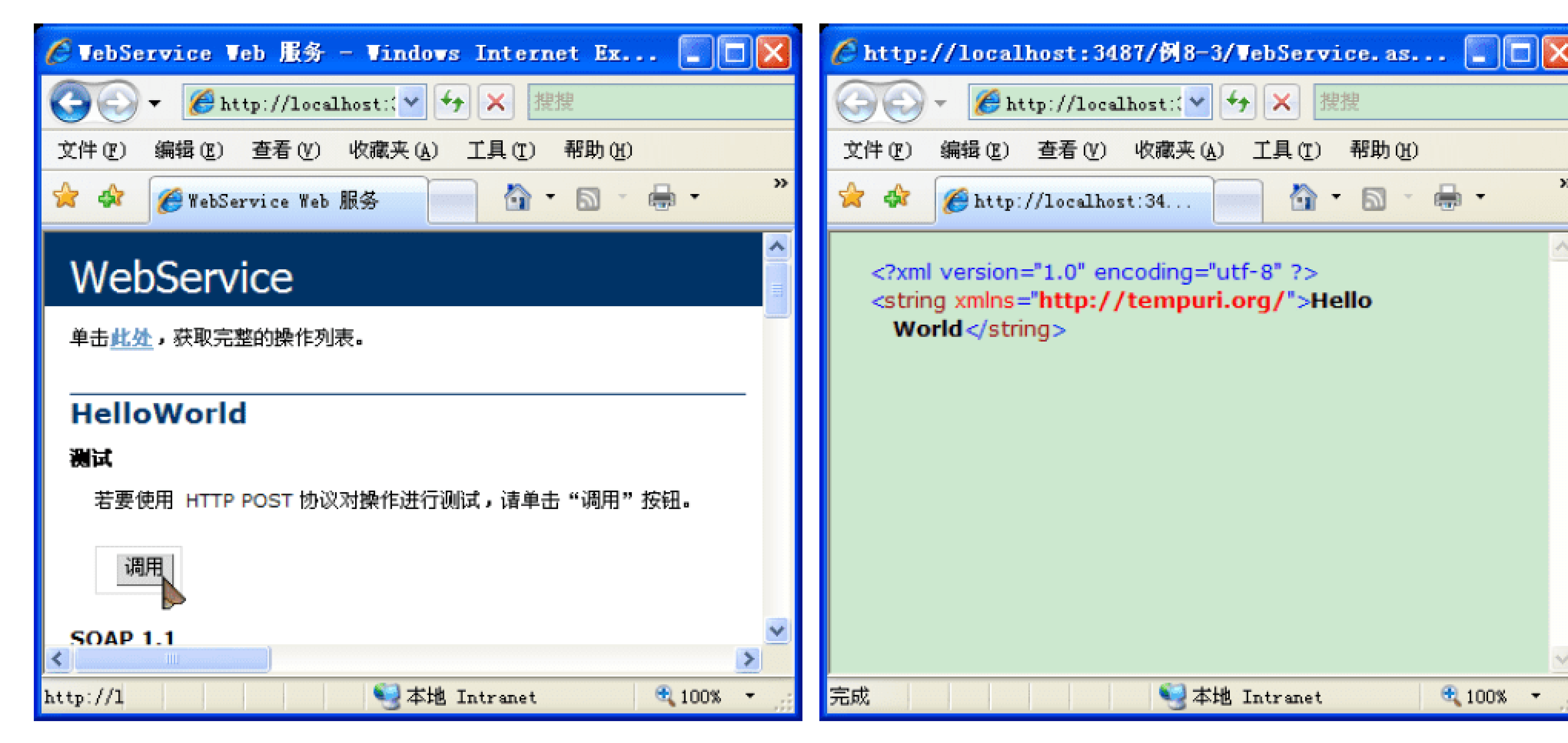


图 8-8　调用 Web 服务　　　　图 8-9　调用 Web 方法返回结果

(8) 返回 WebService.cs 文件，添加绘制验证码的 Web 方法，由于涉及绘图功能，所以在要文件头部引入所需的命名空间，代码如下：

```
using System.Drawing;
using System.IO;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
```

(9) Web 方法的定义如下：

```
[WebMethod]
public byte[] AuthCode(int length,string checkCode)
{
    int bmpHeight = 25, bmpWidth = length * 14 + 10;
    Bitmap bmp = new Bitmap(bmpWidth, bmpHeight);
    int red, blue, green;
    Random rd = new Random(DateTime.Now.GetHashCode());
    red = rd.Next(255) % 128 + 128;
    blue = rd.Next(255) % 128 + 128;
    green = rd.Next(255) % 128 + 128;
    Graphics g = Graphics.FromImage(bmp);
    Brush brush = new SolidBrush(Color.AliceBlue);
    g.FillRectangle(brush,0,0,bmpWidth,bmpHeight);
    //画图片的前景噪音点
    for (int i = 0; i < 30; i++)
    {
        int x = rd.Next(bmpWidth);
```



```
            int y = rd.Next(bmpHeight);
            bmp.SetPixel(x, y, Color.FromArgb(rd.Next()));
        }
        //画图片的边框线
        g.DrawRectangle(new Pen(Color.Silver), 0, 0, bmpWidth - 1, bmpHeight - 1);
        //画图片的背景噪音线
        for (int i = 0; i < 25; i++)
        {
            int x1 = rd.Next(bmpWidth);
            int x2 = rd.Next(bmpWidth);
            int y1 = rd.Next(bmpHeight);
            int y2 = rd.Next(bmpHeight);
            g.DrawLine(new Pen(Color.FromArgb(rd.Next())), x1, y1, x2, y2);
        }
        Rectangle rect = new Rectangle(0, 0, bmpWidth, bmpHeight);
        LinearGradientBrush lgb = new LinearGradientBrush(rect, Color.BlueViolet, Color.DarkRed, 1.2f, true);
        for (int i = 0; i < length; i++)
        {//逐个字符绘制
            Font font = new Font("Courier New", 14+rd.Next()%4, (FontStyle.Bold | FontStyle.Italic));
            g.DrawString(checkCode.Substring(i, 1), font, lgb, 2 + i * 14, 2+rd.Next(2));
        }
        MemoryStream stream = new MemoryStream();
        bmp.Save(stream,ImageFormat.Gif);
        bmp.Dispose();
        g.Dispose();
        byte[] ret = stream.ToArray();//输出字节流
        stream.Close();
        return ret;
    }
```

上述代码通过[WebMethod]属性指定该方法可以被远程调用。随后的代码是绘制验证码的方法，该方法有两个参数，第一个参数是验证码的长度，第二个参数是验证码字符串。

(10) 此时已经完成 Web 服务的创建，可以想刚才那样测试该方法，此时还看不到验证码图片，得到的将是通过 BASE64 编码的一个字符串。

下一节将介绍如何调用 Web 服务，并创建页面调用该方法。

8.4.3 调用 Web 服务

Web 服务的最终目的是提供一种服务接口，由其他程序调用。本节就详细介绍如何调用 Web

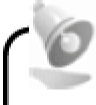


服务，包括调用 Web 服务的机制以及如何调用 Web 服务。

1. 调用 Web 服务的机制

调用 Web 服务的第一步就是先找到一个满足需要的 Web 服务。在找到一个服务后，就可以得到这个 Web 服务的描述信息、分组的分类信息和绑定信息。然后根据描述信息，调用相应的方法。

为了找到已经存在的 Web 服务，Microsoft、IBM 和 Ariba 合作建立了一个带有 UDDI 服务的网站 http://www.uddi.org。如果一个公司要发布自己的 Web 服务，就可以在 UDDI 中注册它。有了 UDDI 商务注册表和 UDDI API，就可以编程定位 Web 服务的信息了。



提示

Web 服务不一定要用 UDDI 注册，也可以从其他资源中获取 Web 服务的信息。

调用 Web 服务的业务流程如图 8-10 所示。

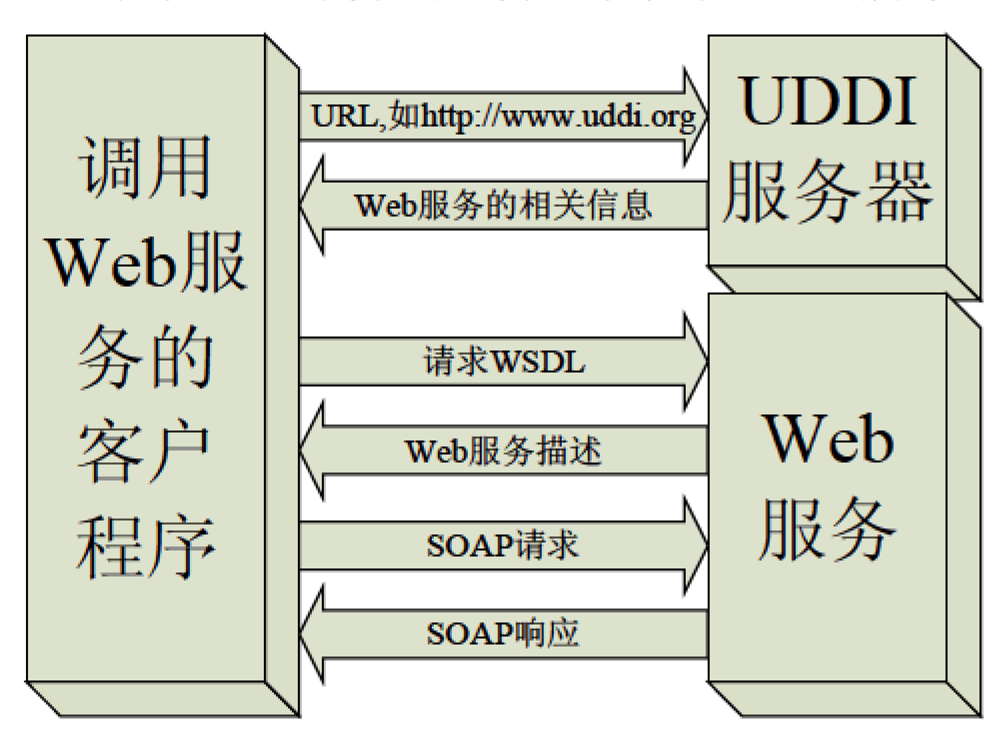


图 8-10　调用 Web 服务的业务流程



知识点

服务的描述信息以 Web Services Description Language (Web 服务描述语言，WSDL)格式显示。WSDL 文档描述了 Web 服务支持什么方法，如何调用这些方法，给服务传送的参数类型，以及调用方法返回的参数类型。在.asmx 文件的最后加上字符串？“wsdl”，就会返回一个 WSDL 文档。WSDL 文档是用 WebMethod 属性动态生成的。

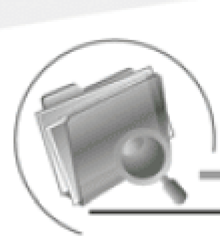
2. 调用 Web 服务

【例 8-4】调用 Web 服务，实现验证码功能。

(1) 启动 VWD 2010，打开网站【例 8-3】。

(2) 在【解决方案资源管理器】面板中，右击【例 8-3】解决方案，从弹出的快捷菜单中选择【添加服务引用】命令，打开【添加服务引用】对话框，单击【发现】按钮即可搜索到上例中创建的 Web 服务，并显示了当前可用的操作，如图 8-11 所示。

(3) 默认的命名空间为 ServiceReference1，稍候编写代码时会用到这个命名空间，单击【确定】按钮将添加 Web 服务引用，网站目录会自动生成一个名为 App_WebReferences 的文件夹，其中包括一个 ServiceReference1(与前面的命名空间名相同)文件夹，里面有 4 个文件，如图 8-12 所示。



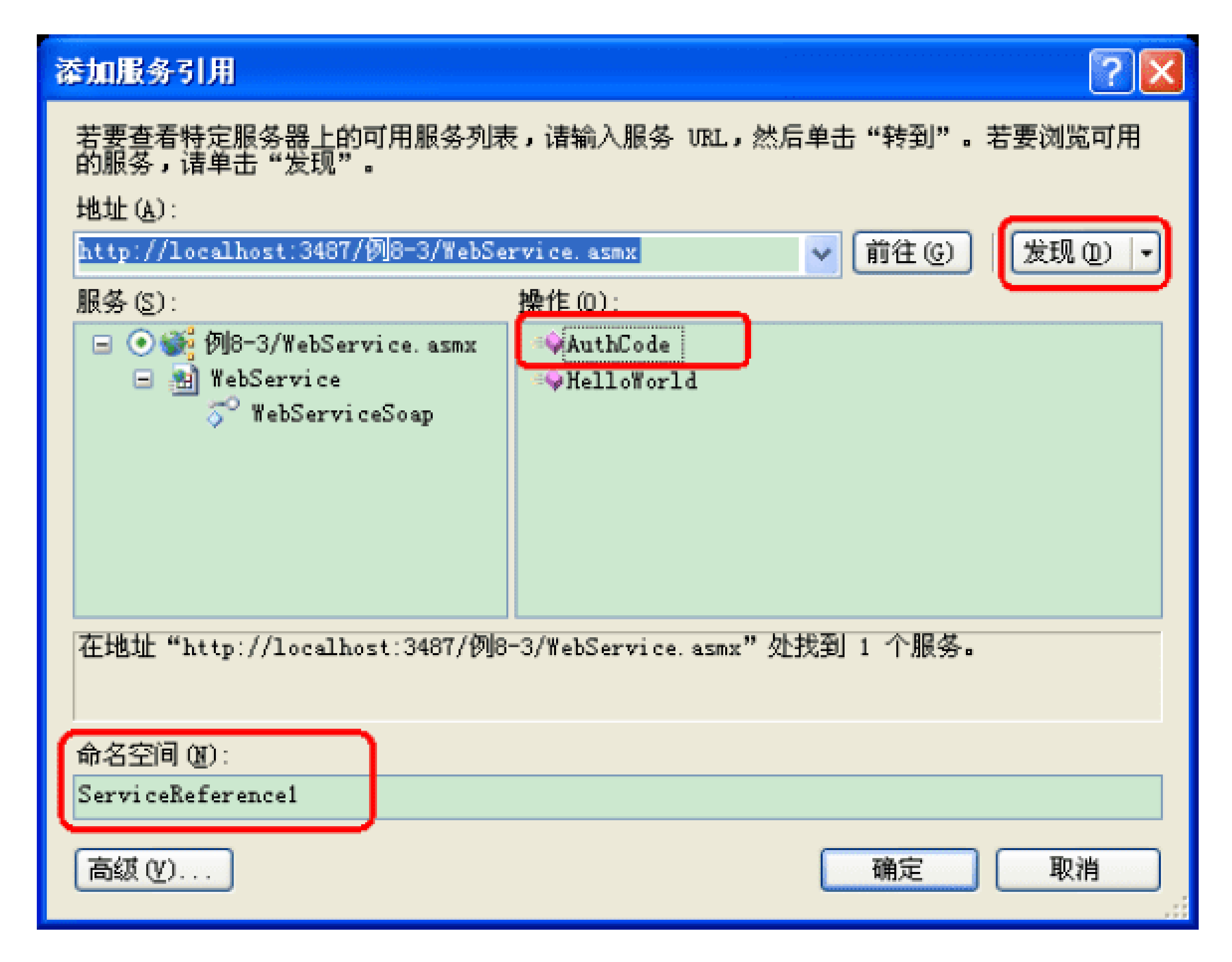


图 8-11 【添加服务引用】对话框

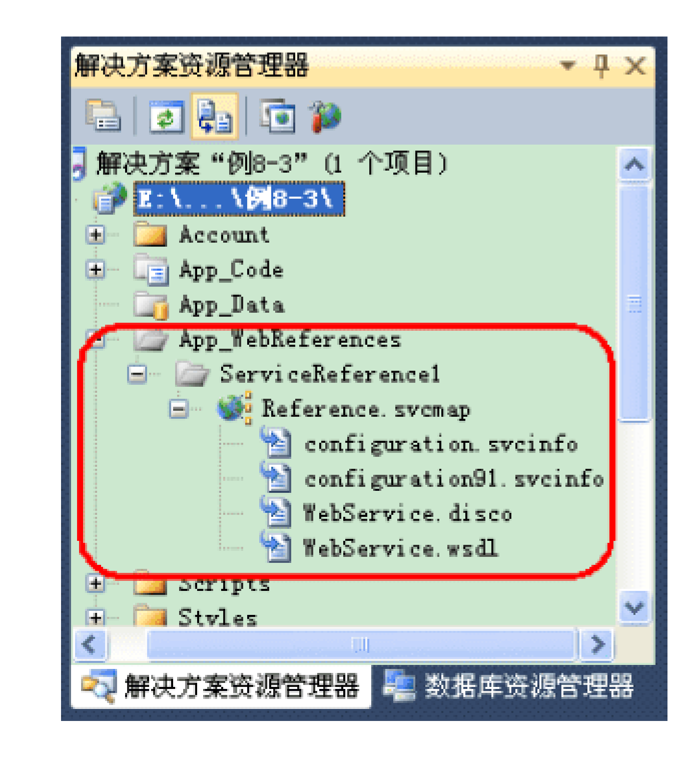
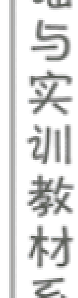


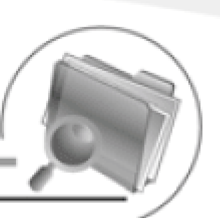
图 8-12 生成的引用文件



(4) 由于 Web 方法 AuthCode 返回的是一个字节数组，为了能将这个字节数组表示的图片显示在 Image 控件中，需要添加一个 Web 页面和一个名为 AuthCode.aspx 的页面。

(5) 在 AuthCode.aspx 的 Load 事件中添加如下代码:

```
protected void Page_Load(object sender, EventArgs e)
{
    ServiceReference1.WebServiceSoapClient c = new ServiceReference1.WebServiceSoapClient();
    byte[] data = c.AuthCode(5,genCode(5));
    Response.ContentType = "image/gif";
    Response.OutputStream.Write(data, 0, data.Length);
}
public string genCode(int length)
{
    char code;
    int number;
    string checkCode = String.Empty;
    Random rd = new Random(DateTime.Now.GetHashCode());
    for (int i = 0; i < length; i++)
    {
        number = rd.Next();
        if (number % 2 == 0)
            code = (char)('0' + (char)(number % 10));
        else
            code = (char)('A' + (char)(number % 26));
```



```
            checkCode += code;
        }
        Session["AuthCode"] = checkCode;
        return checkCode;
    }
```

上述代码先实例化 Web 服务的对象，然后调用 AuthCode 方法获取验证码图片，验证码的字符串信息是通过 genCode 方法生成的，在 genCode 方法最后将生成的验证码字符串保存到 Session 变量中。等用户输入验证码提交后，将从 Session 变量中获得该信息，验证用户的输入。

(6) 打开 Default.aspx 的源代码视图，在<from>标记中添加如下代码：

```
    <form id="form1" runat="server">
    <div>
        验证码：<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:Image ID="Image1" runat="server" ImageUrl="~/AuthCode.aspx" />
        <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Default.aspx">看不清，换一个
</asp:HyperLink>
    </div>
    <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="提交" />
    </form>
```

上述代码中 Image 控件的 ImageUrl 属性指定为 AuthCode.aspx 文件，HyperLink 控件用于刷新页面以便重新获取新的验证码图片。

(7) 在 Button 控件的单击事件处理程序中添加如下代码：

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (Session["AuthCode"].ToString() == TextBox1.Text.Trim().ToUpper())
            Response.Write("<script>alert('验证码输入正确！');</script>");
        else
            Response.Write("<script>alert('验证码输入有误！');</script>");
    }
```

(8) 至此，完成所有代码的编写，编译并运行程序，在浏览器中打开 Default.aspx 页面，如图 8-13 所示。

(9) 输入图片中的字符串，单击【提交】按钮，弹出验证成功对话框，如图 8-14 所示。如果看不清验证码图片，则可以单击“看不清，换一个图片”超链接重新获取验证码，如图输入的验证码有误，也将弹出相应的提示对话框。

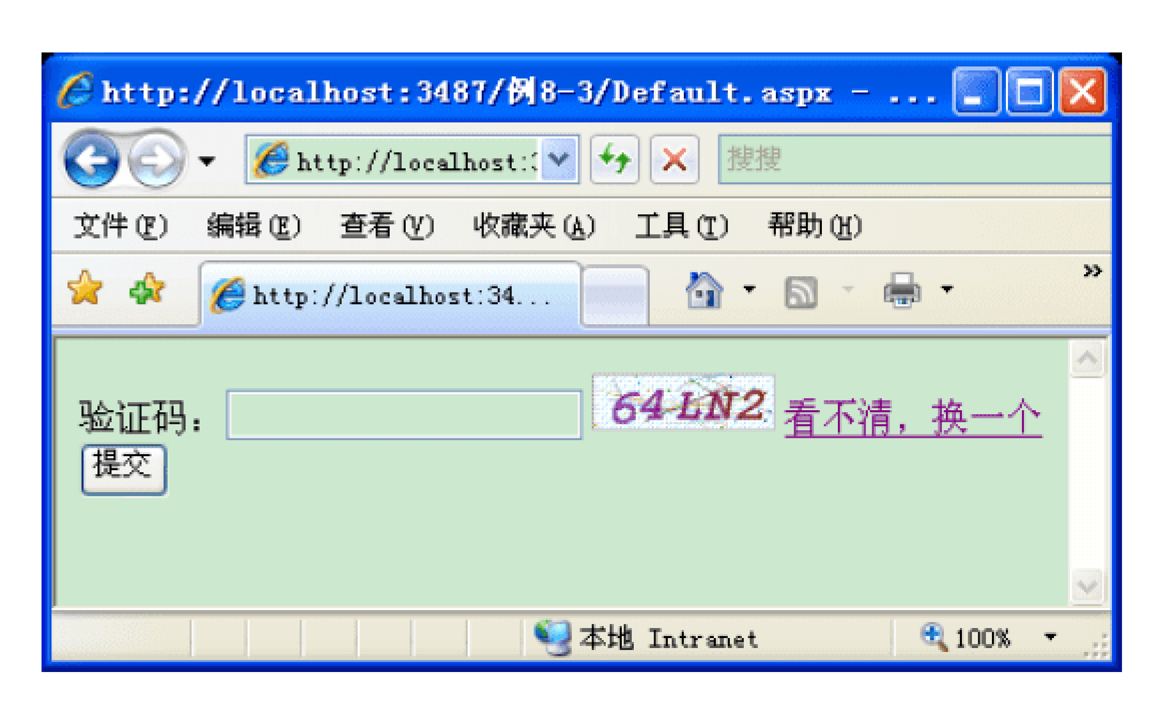


图 8-13　调用 Web 服务显示验证码图片



图 8-14　验证成功对话框

8.4.4　支持 AJAX 的 Web 服务示例

典型的 AJAX 体系结构相当容易理解。其工作原理如图 8-15 所示，其中有一个由应用程序特定服务组成的后端，通常只可调用 AJAX 脚本的外层，其下方是业务逻辑所在和发挥作用的系统中间层。服务与前端通过 HTTP 交换数据，使用多种格式传递参数和返回值。前端由运行于客户端上的 JavaScript 代码组成，在接收和处理完数据后，它面临着使用 HTML 和 JavaScript 构建图形用户界面的重大任务。对 JavaScript 的依赖是由于受浏览器结构的限制，只有当浏览器可以支持功能更加强大的编程功能时，这种情况才会改变。

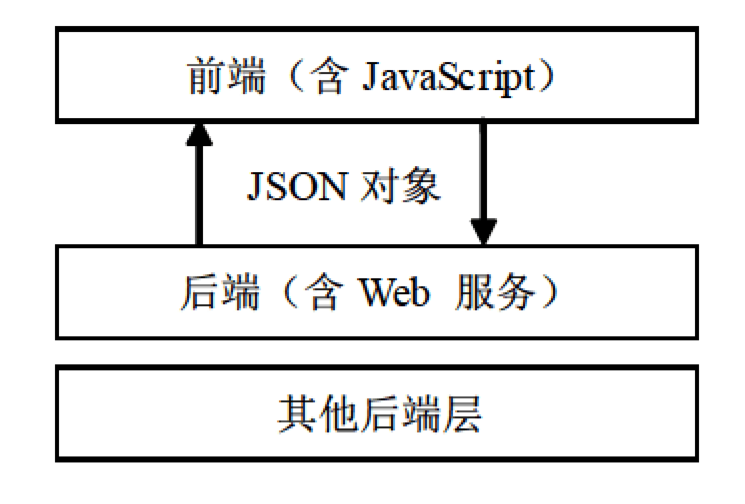


图 8-15　典型的 AJAX 体系结构

知识点

在 AJAX 中，服务表示驻留在应用程序域并向客户端脚本代码公开功能的一段代码。

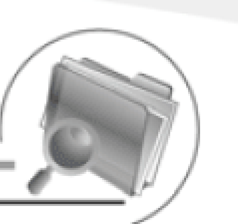
最适合 AJAX 应用程序的服务主要涉及向 Web 客户端公开数据和资源。它可以通过 HTTP 获得，并要求客户端使用 URL(也可以是 HTTP 头)访问数据和命令操作。

1. 创建支持 AJAX 的 Web 服务

前面已经介绍过只需将[System.Web.Script.Services.ScriptService]前面的注释符号删除，即可将整个服务提供为客户端脚本服务。

下面的【例 8-5】将创建这样的服务，在该服务中，添加一个根据学号从数据库中查询学生信息的 Web 方法。

【例 8-5】创建支持 AJAX 的 Web 服务，添加 Web 方法，根据学号从数据库 InfoManage 中查询指定学生的信息。



(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 8-5】。

(2) 在【解决方案资源管理器】面板中，右击【例 8-3】解决方案，从弹出的快捷菜单中选择【添加新项】命令，选择【Web 服务】模板，添加名为 WebService.asmx 的 Web 服务。

(3) 打开 WebService.cs 文件，删除[System.Web.Script.Services.ScriptService]属性前面的注释符号。

(4) 由于要访问数据库，所以需要添加对相应的命名空间的引用：

```
using System.Data.SqlClient;
using System.Data.Sql;
```

(5) 为了方便返回学生信息，我们定义一个结构体 Student，代码如下：

```
public struct Student
{
    public int sno;
    public string sname;
    public string sgender;
    public string stelephone;
    public string saddress;
    public int cno;
}
```



(6) 添加 Web 方法，通过连接数据库查询指定学生的信息，知识点都是以前学过的，此处不做详述，代码如下：

```
[WebMethod]
public Student GetStudentBySno(int sno)
{
    Student stu = new Student();
    string strConnect = "Data Source=zhao\\tsinghua;Initial Catalog=InfoManage;Integrated
Security=True;user=sa;password=Sapassword";
    SqlConnection con = new SqlConnection(strConnect);
    con.Open();
    SqlCommand cmd = new SqlCommand("select * from Student where Sno = @no", con);
    cmd.Parameters.AddWithValue("@no", sno);
    SqlDataReader reader = cmd.ExecuteReader();
    if (reader.Read())//获取查询结果
    {
        stu.sno = reader.GetInt32(0);
        stu.sname = reader.GetString(1);
```

```
                stu.sgender = reader.GetString(2);
                stu.stelephone = reader.GetString(3);
                stu.saddress = reader.GetString(4);
                stu.cno = reader.GetInt32(5);
            }
            cmd = null;
            con.Close();
            con = null;
            return stu;
        }
```

(7) 至此已完成 Web 服务的创建，可以在浏览器中加载并测试该服务。

2. 在 AJAX 站点中使用 Web 服务

在第 7 章介绍过，ScriptManager 控件几乎是所有与 AJAX 相关的操作中必不可少的。它注册客户端 JavaScript 文件，负责使用 UpdatePanel 更新部分页面，可以将 ScriptManager 添加到单个页面或母版页中，让它变得在整个站点上都可用。

使用 Web 服务时，需要告知 ScriptManager 要给客户端脚本提供 Web 服务。有两种方法可以实现：

- 在母版页中的 ScriptManager 中。
- 在使用 Web 服务的内容页中使用 ScriptManagerProxy 控件。

要在全部或大多数页面中使用 Web 服务，最好是在母版页的 ScriptManager 中声明 Web 服务。给 ScriptManager 控件提供一个<Services>元素，该元素再包含指向公共服务的一个或多个 ServiceReference 元素。

通过在母版页中引用 Web 服务，它变得对于基于这个母版页的所有页面都可用。这也意味着每个页面都要下载运行这个服务所需的 JavaScript 文件。如果页面根本没有使用 Web 服务，这也会浪费带宽和资源。因此，对于只在一些页面上使用的服务，最好引用页面本身的服务。如果使用的母版页有 ScriptManager 控件，那么在内容页中就要使用 ScriptManagerProxy 控件。本例中就介绍如何在内容页中使用 ScriptManagerProxy 控件来注册 Web 服务。

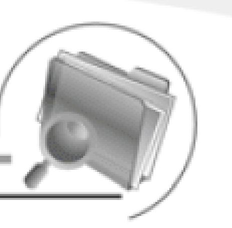
【例 8-6】在 AJAX 站点中调用 Web 服务。

(1) 启动 VWD 2010，打开网站【例 8-5】。

(2) 在【解决方案资源管理器】面板中，右击【例 8-3】解决方案，从弹出的快捷菜单中选择【添加新项】命令，添加一个母版页 MasterPage.master。

(3) 在母版页中添加 ScriptManager 控件。需要注意的是，ScriptManager 控件要添加在 ContentPlaceHolder 控件的外部。

(4) 删除原来的 Default.aspx 页面，基于刚才创建的母版页 MasterPage.master 重新创建 Default.aspx 页面。



(5) 在 Default.aspx 页面中添加一个 ScriptManagerProxy 控件，在【属性】面板中设置控件的 Services 属性，这是一个集合属性，单击属性右侧的省略号将打开【ServiceReference 集合编辑器】对话框。单击【添加】按钮，添加一个成员，设置 Path 值为前面创建的 Web 服务，如图 8-16 所示。切换到源视图，生成的代码如下：

```
<asp:ScriptManagerProxy ID="ScriptManagerProxy1" runat="server">
    <Services>
        <asp:ServiceReference Path="~/WebService.asmx" />
    </Services>
</asp:ScriptManagerProxy>
```

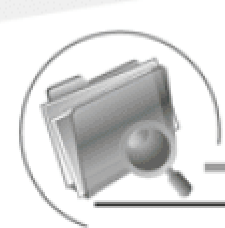
(6) 在<ScriptManagerProxy>的结束标记下方，添加一个 Input (Text)和一个 Input (Button)，在输入相应的文本提示信息，然后添加一个<div>标记，用于显示返回信息。

```
按学号查询学生信息：<input id="Text1" type="text" />
<input id="Button1" type="button" value="查询" />
<div id="resu"></div>
```

(7) 在上述代码的下面添加客户端 JavaScript 代码块：

```
<script type="text/javascript">
    function GetWebMethod() {
        WebService.GetStudentBySno($get('Text1').value, onSuccCallback,onFailCallback);
    }
    function onSuccCallback(result) {
        var info;
        info = "学号：" + result.sno;
        info += "<br>姓名：" + result.sname;
        info += "<br>性别：" + result.sgender;
        info += "<br>联系电话：" + result.stelephone;
        info += "<br>地址：" + result.saddress;
        info += "<br>班级号：" + result.sno;
        $get('resu').innerHTML = info;
    }
    function onFailCallback(error) {
        alert(error.toString);
    }
    $addHandler($get('Button1'), 'click', GetWebMethod);
</script>
```

(8) 编译并运行程序，输入一个学号，然后单击【查询】按钮，如图 8-17 所示。





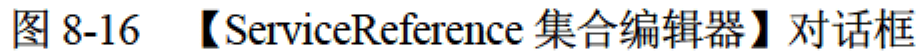


图 8-16 【ServiceReference 集合编辑器】对话框

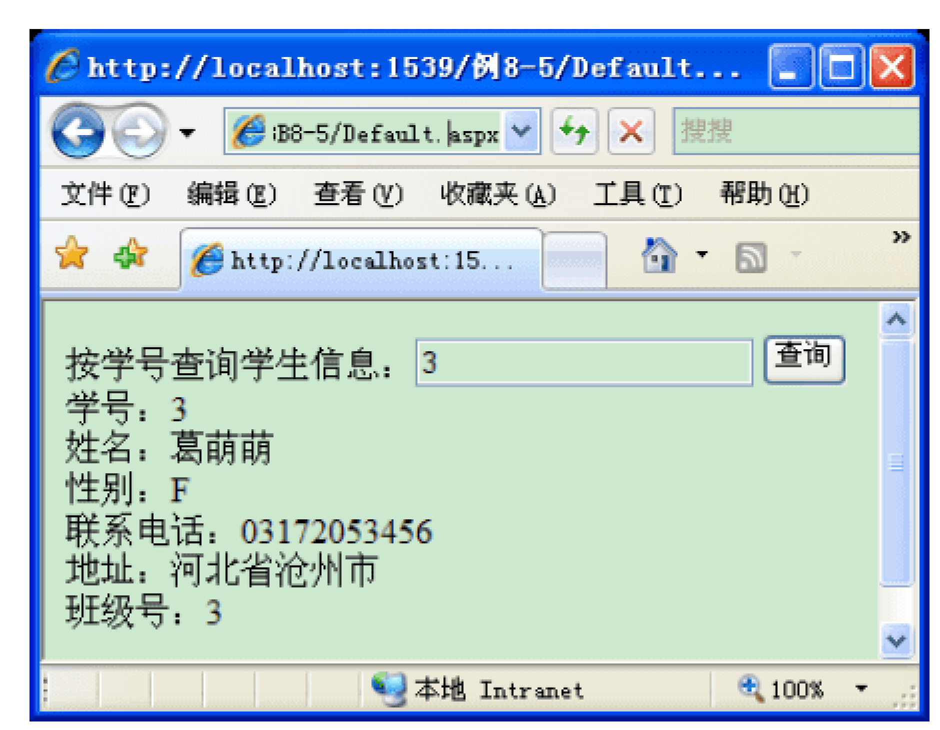


图 8-17 页面运行效果



8.5 上机练习

本章的上机练习将演示如何在 AJAX 站点中调用页面方法。

页面方法和 Web 服务有一些共同之处。两者都可以使用很少的代码在客户端调用。可以向它们发送数据，并接收回发的数据。另外，当调用它们时，可以定义成功和失败回调方法。两者的不同之处在于，页面方法直接在现有的 ASPX 页面内定义，而不是在单独的 ASMX 服务文件中定义。只能从页面运行的脚本中调用页面方法。

要启用页面方法，需要将 ScriptManager 控件的 EnablePageMethods 属性设置为 True。不能在 ScriptManagerProxy 类上设置该属性，因此需要直接在 ScriptManager 控件上进行设置。

(1) 启动 VWD 2010，新建网站【上机练习 8】。

(2) 在 Default.aspx 页面中添加一个 ScriptManager 控件，设置控件的 EnablePageMethods 属性为 True。

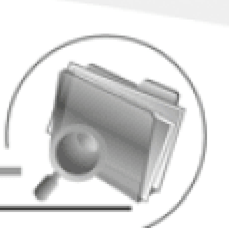
(3) 打开 Default.aspx.cs 文件，添加如下页面方法：

```
[WebMethod]
public static string HelloWorld(string yourName)
{
    return string.Format("Hello {0}", yourName);
}
```

在添加上述代码之前需要首先引入所需的命名空间：using System.Web.Services;

(4) 切换到 Default.aspx 页面的源视图，在</ScriptManager>标记下面添加一个 Input(text)和一个 Input (Button)控件，方法是从工具箱的 HTML 类别中拖动它们。将按钮的 value 设置为“提交”。相应的代码如下：

```
<input id="Text1" type="text" />
<input id="Button1" type="button" value="提交" />
```



(5) 接下来，添加如下代码：

```
<script type="text/javascript">
$addHandler($get('Button1'), 'click', HelloPageMethod);
function HelloPageMethod(){
    var name = $get('Text1').value;
    PageMethods.HelloWorld(name, HelloCallback);
}
function HelloCallback(result) {
        alert(result);
}
</script>
```

(6) 编译并运行程序，输入一个用户名，单击【提交】按钮，如图 8-18 所示。

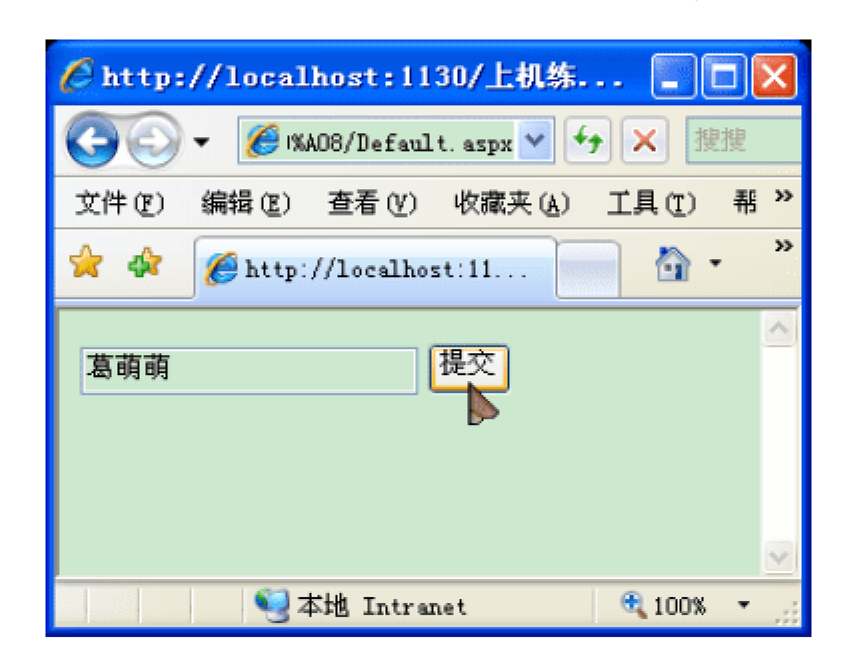




图 8-18　调用页面方法允许效果



8.6　习题

1. XML 有哪些具体应用？
2. 简述调用 Web 服务的机制和工作原理。
3. 如何创建支持 AJAX 的 Web 服务？
4. 调用天气预报 Web 服务获取国内大城市的天气信息。
5. 创建一个支持 AJAX 的 Web 服务。
6. 登录 http://www.onhap.com/，选择自己喜欢的 Web 服务，并进行编程练习。



第9章 jQuery 入门

学习目标

前面的章节中介绍了 JavaScript，这是在客户端编写脚本和与客户端的 Web 页面元素进行交互所运用的实际标准语言。jQuery 是继 Prototype 之后出现的又一个优秀的 JavaScript 框架。jQuery 能够改变开发人员编写 JavaScript 脚本的方式，降低学习和使用 Web 前端开发的复杂度，提高网页开发效率。无论是对于 JavaScript 初学者，还是对于 Web 开发资深专家，jQuery 都是必备的工具。本章主要介绍 jQuery 的基本语法和具体应用。

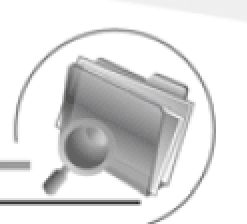
本章重点

- jQuery 简介
- jQuery 选择器
- jQuery 筛选器
- 使用 jQuery 增强页面
- 使用 jQuery 插件
- jQuery 与 AJAX

9.1 jQuery 简介

jQuery 最早由 John Resig 在 2006 年 1 月发布，现在已经成长为一个备受欢迎的客户端框架。Microsoft 也注意到了 jQuery 的强大功能，并决定在自己的产品中附送这个框架。最初，jQuery 随 Microsoft ASP.NET MVC 框架一起提供，现在 Visual Studio 和 Visual Web Developer 2010 中也包含了这个框架。

jQuery 库的主要关注点一直是简化访问 Web 页面元素的方法、帮助处理客户端事件、提供视觉效果(如动画)支持，以及使得在应用程序中使用 AJAX 变得更加简单。2006 年 1 月，John Resig



公布了 jQuery 的第一版，然后在 2006 年 8 月正式发布了 jQuery 1.0。后来又陆续发布了许多版本，目前最新的稳定版本是 jQuery 1.4.1。

使用 ASP.NET Web 站点模板创建的任意新 Web 站点都包含一个 Scripts 文件夹，其中已经包含了必要的 jQuery 文件。如果基于 ASP.NET 空 Web 站点模板建立网站，也可以手动向 Web 站点添加 jQuery 文件。

因为 jQuery 库会增加网页的大小，所以应该明确决定是否在 Web 站点中包含它。当决定把 jQuery 库添加到 Web 站点中时，还需要作出一些选择。

9.1.1　选择引用 jQuery 的位置

要在 Web 站点中包含 jQuery，有几种选项可供选择：

- 只在需要 jQuery 的网页或者用户控件中添加对 jQuery 库的引用。这种方式可以减小页面大小。当用户浏览没有使用 jQuery 的页面时，就不需要下载 jQuery 库文件。而当下载了库文件之后，浏览器就会缓存库文件的一个副本，从而使得在以后访问页面时，不需要再次下载这些文件。
- 在 Web 站点的母版页中添加对 jQuery 库的引用，从而使所有的页面都可以使用 jQuery 库。这种方式十分方便，因为所有基于该母版页创建的页面都会自动获得对 jQuery 的访问权。但是，这会对 Web 站点第一个页面的性能造成冲击，因为需要从服务器上下载库文件。

因为 jQuery 库很小，所以一般是在母版页中包含它。



9.1.2　包含 jQuery 库的不同方式

因为 jQuery 库由一个使用 JavaScript 代码编写的文件组成，所以可以使用标准的<script>语法在页面、用户控件或母版页中嵌入对 jQuery 库的引用，例如：

```
<script src=" jquery-1.4.1.min.js" type="text/javascript"></script>
```

必须使用一个独立的结束</script>标记，因为如果使用自结束标记，一些浏览器将无法正常运行代码。

也可以将引用嵌入到 ScriptManager 控件中。ScriptManager 控件有一个<Scripts>子元素，可以用来注册将会添加到浏览器的最后一个页面的 JavaScript 文件。在 ScriptManager 中注册 JavaScript 文件的最简形式如下所示：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
  <Scripts>
    <asp:ScriptReference Path="~/Scripts/jquery-1.4.1.min.js" />
```

```
    </Scripts>
</asp:ScriptManager>
```

另外一种方法是使用 Microsoft 的内容传送网络(Content Delivery Network，CDN)或 Google Code 引用 jQuery 库的在线版本。

使用外部库的在线版本的优势在于可以提升服务器的性能并降低带宽。因为站点的访问者很可能已经在访问另外一个站点的时候下载了共享脚本。

9.1.3 第一个 jQuery 页面

为了更好地了解 jQuery，下面先来看一个简单的例子。在本例中，将在当前页面中添加 jQuery 库。通过单击按钮来改变表单的背景色。

【例 9-1】使用 jQuery 示例。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 9-1】。

(2) 通过 VWD 2010 创建的网站，默认有一个 Scripts 目录，其中包含了 jQuery 所需的库文件。切换到 Default.aspx 的源视图，在<head>标记中添加如下代码即可引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 在<div>标记中添加两个 Input(Button)控件，该控件的 value 属性分别为“红色”和“蓝色”。生成的代码如下：

```
<input id="Button1" type="button" value="红色" />
<input id="Button2" type="button" value="蓝色" />
```

(4) 在上述代码的下面添加如下代码：

```
<script type="text/javascript">
    $(document).ready(function () {
        $('#form1').css('background-color', 'green')
        $('#Button1').click(function () {
            $('#form1').css('background-color', 'red')
                    .animate({ width: '300px', height: '200px' })
        });
        $('#Button2').click(function () {
            $('#form1').css('background-color', 'blue')
                    .animate({ width: '350px', height: '150px' })
        });
    });
</script>
```



提示

和其他许多编程语言一样，JavaScript(以及 jQuery)对缺少引号、大括号和小括号十分敏感，所以一定要完全按照上面的代码进行输入。在输入代码时，【智能感知】将会弹出，并通过工具提示提供关于各种方法和参数的信息。如果没有弹出【智能感知】，那么可能是没有正确添加<script>元素。

(5) 编译并运行程序，在默认浏览器中打开 Default.aspx 页面，如图 9-1 所示，表单背景色为绿色，而且仅为一长条。

(6) 单击【红色】按钮，将把背景色设置为红色，大小为(300,200)，如图 9-2 所示。

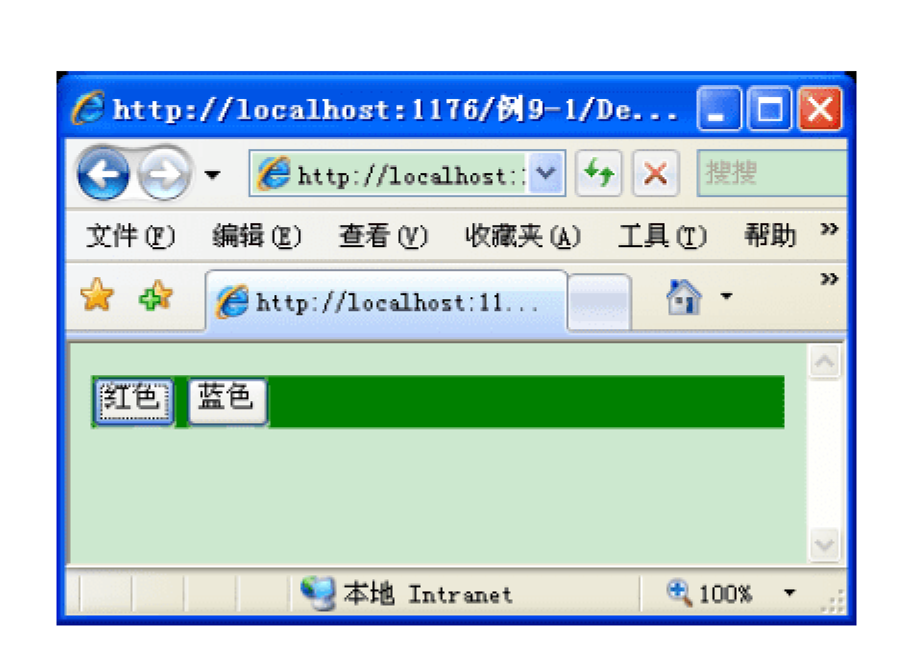




图 9-1　初次加载页面效果

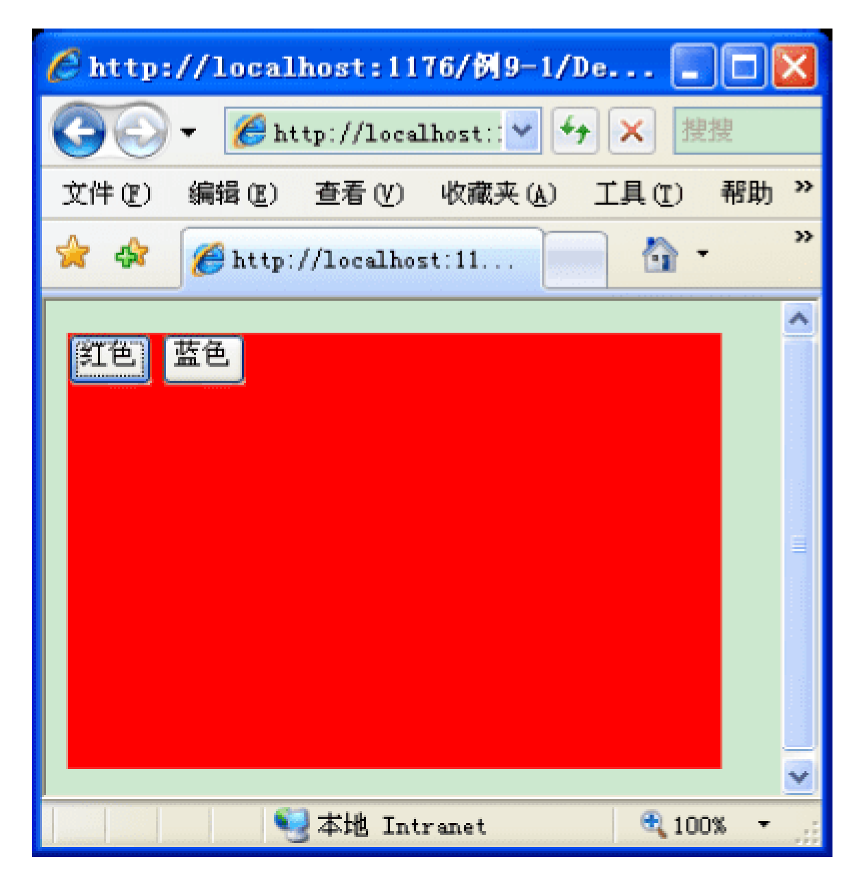


图 9-2　单击【红色】按钮后的效果



上述代码中，添加了一个标准的<script>块，其中可以包含 JavaScript。在这个块中，添加了一些在浏览器加载页面完成后触发的 jQuery 代码。页面就绪后，起始大括号({)和结束大括号()) 之间的代码将会执行，下面是“文档就绪函数”的示例。

```
<script type="text/javascript">
  $(document).ready(function() {
    // Remainder of the code skipped
  });
</script>
```

本例读者只需了解 jQuery 的实际应用即可，接下来将详细介绍 jQuery 的语法。

9.2　jQuery 语法

要想理解和使用 jQuery，需要掌握一些基础知识。本节将介绍 jQuery 的核心功能，包括前面看到的$函数，以及$函数的 ready 方法。接下来介绍 jQuery 的选择器和筛选器，这样就可以通

过自己指定的条件在页面中查找元素。当获得一个指向页面中一个或多个元素的引用后，就可以对它们应用多种方法，如前面提到的 css 方法。

9.2.1 ready 函数

大部分 jQuery 代码都是在浏览器完成页面加载后执行。等到页面完成 DOM 加载后再执行代码十分重要。DOM(Document Object Model，文档对象模型)是 Web 页面的一种分层表示，包含所有 HTML 元素、脚本文件、CSS、图像等的一个树形结构。如果借助编程修改 DOM(例如，使用 jQuery 代码)，那么这种修改将在浏览器中显示的页面上反映出来。如果过早执行 jQuery 代码(例如，在页面的最顶端)，那么 DOM 可能还没有加载完成脚本中引用的全部元素时就产生了错误。幸运的是，可以使用 jQuery 中的 ready 函数，将代码的执行推迟到 DOM 就绪。

ready 函数的声明格式如下：


```
$(document).ready(function() {
  // DOM 就绪后执行此处的代码
});
```

当页面准备就绪，可以执行 DOM 操作时，添加到起始和结束大括号之间的全部代码都将执行。jQuery 也提供了 ready 函数的一个快捷方式，下面的代码段与前面的效果相同：

```
$(function() {
  // DOM 就绪后执行此处的代码
});
```

9.2.2 基本选择器

在 jQuery 中，可以使用美元符号($)作为在页面中查找元素的快捷方式，找到并返回的元素称为匹配集。$方法的基本语法如下所示：

```
$('选择器')
```

在引号(可以使用单引号或者双引号，只要在两端使用相同的类型即可)之间，可输入一个或多个选择器，接下来就将讨论这方面的内容。

通过 jQuery 选择器可以找到页面的文档对象模型中的一个或多个元素，以便向它们应用各种类型的 jQuery 方法。jQuery 的设计者并没有开发出一种新技术来查找页面元素，而是使用与 CSS 选择器完全相同的选择器。

1. 通用选择器

和对应的 CSS 选择器一样，通用选择器使用通配符*匹配页面中的全部元素；$方法返回 0



个或多个元素，然后可以使用多种 jQuery 方法操作返回的这些元素。例如，要将页面中每个元素的字体系列设置为 Arial，可以使用下面的代码：

```
$('*').css('font-family', 'Arial');
```

2. ID 选择器

和对应的 CSS 选择器一样，这个选择器通过 id 来查找和获取元素。例如，要为名为 tabel1 的表格设置 CSS 类，可以使用如下代码：

```
$('#table1').addClass('myClass');
```

当这行代码使用 addClass 方法设置 CSS 类时，将会遵循标准的 CSS 规则，即需要通过外部 CSS 文件或者嵌入式样式表定义 myClass 类。

jQuery 的$('#table1')和 ASP.NET AJAX 的$get('table1')都会获得对 id 为 table1 的一个元素的引用。那么应该选择哪一个方法呢？一般来说，当对结果应用任意 jQuery 方法(例如 css 方法)时，都应该使用 jQuery 的$方法。而在操作单个元素并且想要修改该元素的某个标准属性时，则可以使用$get 代替。在这种情况下，也可以使用 jQuery 的$，但是因为所有的 jQuery 选择器都返回一个对象集合，所以需要通过索引方式，使用[0]或 get(0)得到第一个元素。下面的 3 个示例具有相同的功能，它们都将 Button1 的值设置为“单击”：

```
$get('Button1').value = '单击';
$('#Button1')[0].value = '单击';
$('#Button1').get(0).value = '单击';
```

3. 元素选择器

元素选择器获得与特定的标记名相匹配的 0 个或多个元素的引用。例如，下面的代码将页面中的所有二级标题的文本颜色设置为红色。

```
$('h2').css('color', 'red');
```

4. 类选择器

类选择器获得与特定的类名相匹配的 0 个或多个元素的引用。例如，如果有下面的 HTML 代码段：

```
<h1 class="Highlight">Heading 1</h1>
<h2>Heading 2</h2>
<p class="Highlight">First paragraph</p>
<p>Second paragraph</p>
```

上述 4 个元素中有两个元素都有一个名为 Highlight 的 CSS 类。下面的 jQuery 代码将把第一

个标题和第一个段落的背景色修改为红色，而保持其他元素不变：

```
$('.Highlight').css('background-color', 'red');
```

5. 分组和合并选择器

和 CSS 一样，可以分组和合并选择器。下面的分组选择器将修改页面中所有 h1 和 h2 元素的文本颜色为橙色：

```
$('h1, h2').css('color', 'orange');
```

通过使用合并选择器，可以找出被其他一些元素包含着的特定元素。例如，下面的 jQuery 只修改 MainContent 元素中包含的二级标题，而其他的保持不变：

```
$('#MainContent h2').css('color', 'red');
```



6. 层级选择

jQuery 支持 4 类层级选择器，分别如下：

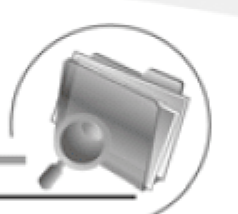
- ancestor descendant：在指定祖先元素下匹配所有的后代元素，与 CSS 中的包含选择器对应。
- parent > child：在给定的父元素下匹配所有的子元素，与 CSS 中的子选择器对应。
- prev + next：匹配所有紧接在 prev 元素后的 next 元素，与 CSS 中的相邻选择器对应。
- prev ~ siblings：匹配 prev 元素之后的所有 siblings 元素。

假设有下面的代码：

```
<form>
  <label>Name:</label>
  <input name="name" />
  <fieldset>
      <label>Newsletter:</label>
      <input name="newsletter" />
 </fieldset>
</form>
<input name="none" />
```

使用层级选择器的 jQuery 代码如下：

```
$("form input")   //返回结果: <input name="name" />, <input name="newsletter" />
$("form > input")   //返回结果: <input name="name" />
$("label + input")   //返回结果: <input name="name" />, <input name="newsletter" />
$("form ~ input")   //返回结果: <input name="none" />
```



7. 使用选择器

为了理解 jQuery 选择器以及可以对匹配集应用的效果，下面举例来说明如何使用一些选择器，以及如何对匹配集应用动画。

【例 9-2】使用 jQuery 选择器。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 9-2】。

(2) 切换到 Default.aspx 的源视图，在<head>标记中添加如下代码以引入 jQuery 库:

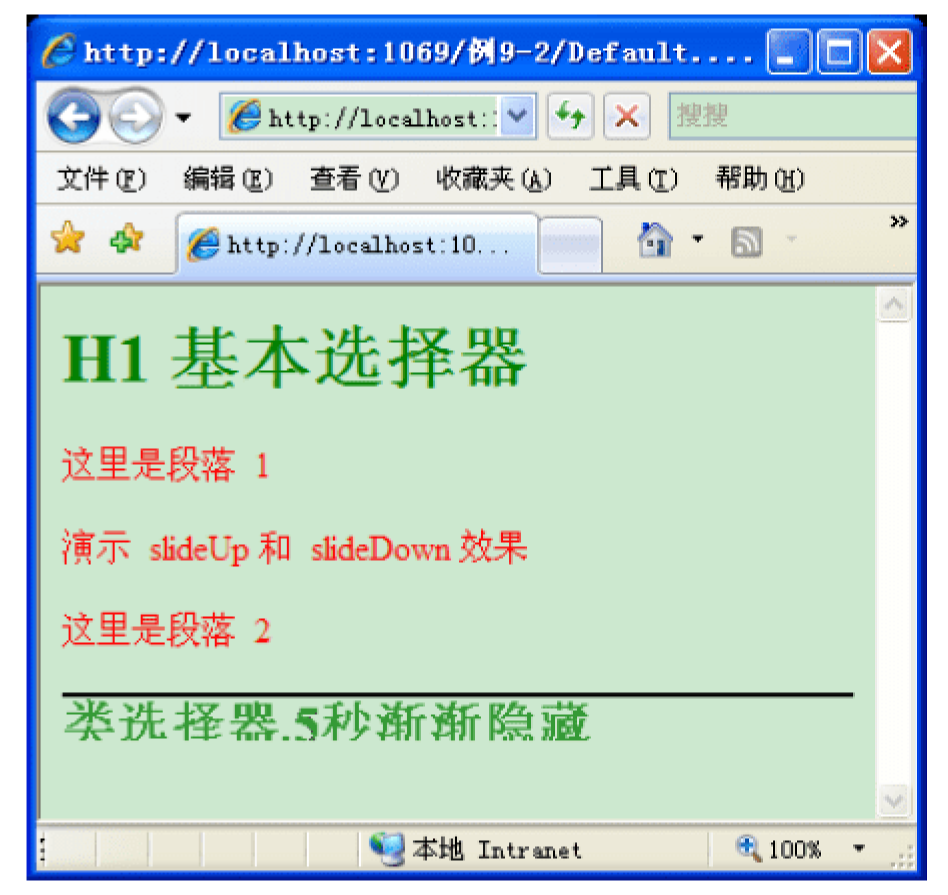
```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 在<form>标记中添加如下代码:

```
<form id="form1" runat="server">
<h1>H1 基本选择器</h1>
<div>
<p>这里是段落 1</p>
<div id="slide">演示 slideUp 和 slideDown 效果
<p>这里是段落 2</p></div>
<h2 class="SampleClass">类选择器,5 秒渐渐隐藏</h2>
<script type="text/javascript">
    $(function () {
        $('*').css('color', 'Green');
        $('#slide').css('border-bottom', '2px solid black');
        $('h1').bind('click', function () { alert('Hello 葛萌萌') });
        $('.SampleClass').hide(5000);
        $('#slide,p').css('color', 'red');
        $('#slide p').slideUp('slow').slideDown('slow');
    });
    </script>
</div>
</form>
```

(4) 上述 jQuery 代码部分，首先设置所有文本的颜色为绿色，设置 slide 层的下方有一个额外的边框，接着为<h1>元素绑定一个 click 函数，当单击该元素时，将弹出一个对话框，然后通过类选择器设置<h2>元素的隐藏效果，随后又通过分组选择器将 slide 和 p 的文本颜色设置为红色，最后设置 slide 中的 p 为淡入淡出动画效果。编译并运行程序，在浏览器中打开 Default.aspx 页面。运行效果如图 9-3 所示。





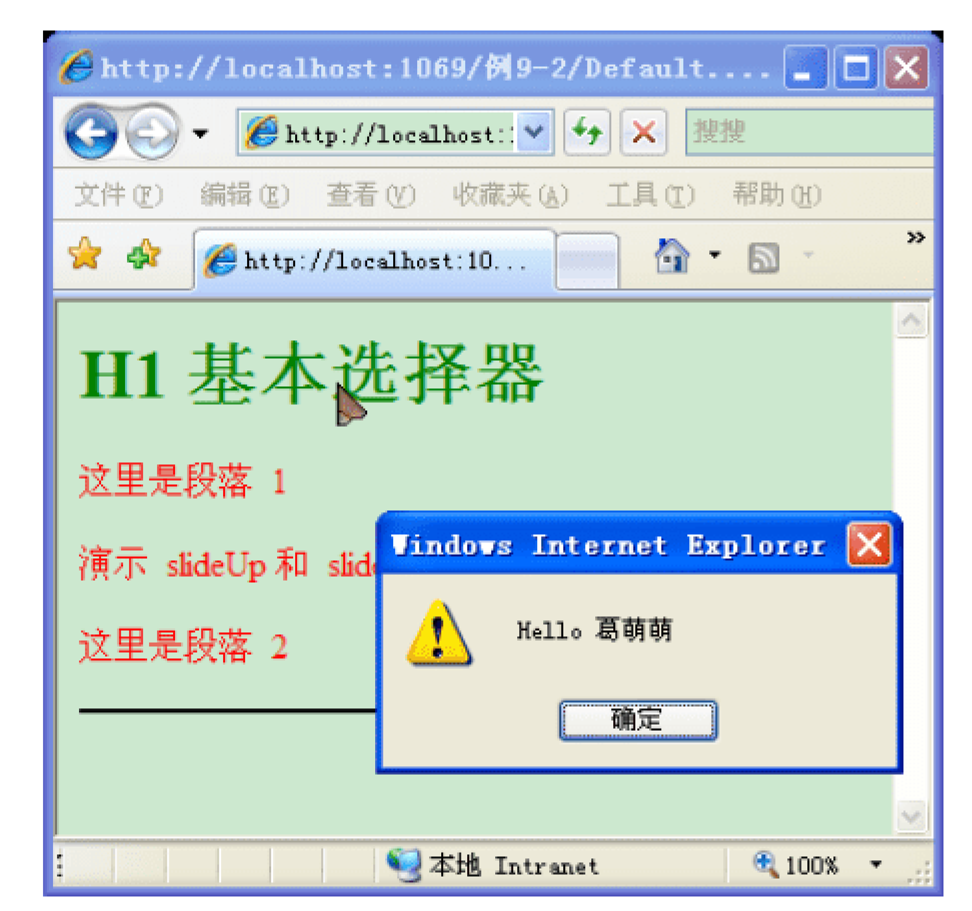


图 9-3　使用选择器页面效果



提示

添加大量动画会使页面看上去很有趣，但是一般不推荐将动画功能添加到页面中，但是，本例用动画作为演示效果却很好，因为可以从中看到 jQuery 的一些强大功能。

在本章后面的部分中，将会介绍 jQuery 提供的更多样式和动画方法。现在只需要理解选择器语法，能够在页面中引用元素即可。

9.2.3　筛选器

在 jQuery 中，可以使用筛选器进一步过滤选择器得到的结果集，从而可以找到特定的元素，如第一个元素、最后一个元素、所有奇数行元素、所有偶数行元素、所有的标题或者特定位置的项。

1. 基本筛选器

如表 9-1 所示列出了 jQuery 的基本筛选器。

表 9-1　基本筛选器

筛 选 器	用　　途
:first :last	用于选择匹配集中的第一个和最后一个项。下面的示例将表的第一行和最后一行的背景色设置为红色： $('#TableId tr:first').css('background-color', 'red'); $('#TableId tr:last').css('background-color', 'red'); 首先使用#TableId 找到表，然后使用 tr 找到表的全部行，最后使用:first 和:last 筛选器找到第一行和最后一行

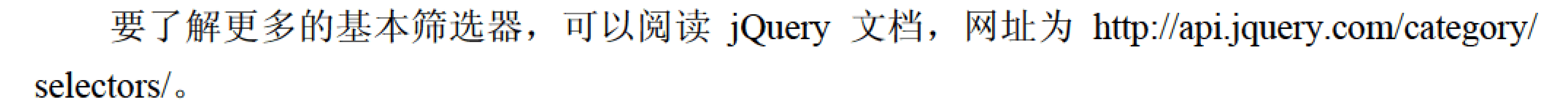
(续表)

筛 选 器	用　　途
:odd :even	用于选择匹配集中的奇数行或者偶数行。下面的示例将表的奇数行的背景色修改为红色。因为计数是从零开始的，所以实际上将会看到第二行和第四行的背景色发生了改变(因为它们的索引分别为 1 和 3) $('#TableId tr:odd').css('background-color', 'red');

:eq(index)
:lt(index)
:gt(index)
	按照索引匹配元素。:eq(equals)根据索引返回一个元素，而:lt(less than)和:gt(greater than)则分别返回小于或者大于给定索引的项。示例如下：
	$('#TableId tr:eq(0)').css('color', 'green');//修改第一行的颜色
	$('#TableId tr:gt(2)').css('color', 'green');//修改大于第二行的文本颜色
	$('#TableId tr:lt(2)').css('color', 'green');//修改行号小于 2 的行的文本颜色

:header	找到页面中的全部标题(从 h1 到 h6)。示例如下所示： $(':header').css('color', 'green');





要了解更多的基本筛选器，可以阅读 jQuery 文档，网址为 http://api.jquery.com/category/selectors/。

2. 高级筛选器

除了刚才看到的基本筛选器以外，jQuery 还支持其他很多筛选器，它们可以用来根据项包含的文本、是否可见、以及它们包含的任意属性获取项。另外，还有一些筛选器可以获得表单元素(例如按钮、复选框、单选按钮等)，以及大量可以用来选择子元素、父元素、兄弟元素和后代元素的选择器。如表 9-2 所示列出了最常用的筛选器。

表 9-2　高级筛选器

筛 选 器	用　　途
:contains(text)	通过包含的文本匹配元素。示例如下： $('td:contains("Row 3")').css('color', 'green'); 如果省略 td，那么整个表都会变成绿色。这是因为表本身也会被匹配(它的一个子表包含文本 Row 3)，所以颜色将应用到整个表上，从而使得每个单元格的文本变为绿色
:has(element)	匹配至少包含一个给出元素的元素。示例如下： $(':header:has("span")').css('color', 'green');//将包含 span 的标题元素颜色修改为绿色
[attribute]	基于给定属性匹配元素。示例如下： $('[type]').css('color', 'green'); 需要在文本框中输入一些文本来查看绿色的字体
[attribute=value]	基于一个属性和该属性的值匹配元素。示例如下： $('[type=text]').css('color', 'green');

(续表)

筛 选 器	用 途
:input :text :password :radio :checkbox :submit :image :reset :button :hidden :file	这些选择器可以用来匹配特定的客户端 HTML 表单元素。例如，可以使用分组选择器把查找按钮和文本框的代码段重写如下： $(':button, :text').css('color', 'green'); 可以使用这些筛选器来实现一些特殊的效果。例如，要想编写一些功能来选中一个表单中的所有复选框，可以使用下面的代码： $(':checkbox').attr('checked', true); 要想取消全部复选框，可以传递 false 作为 attr 方法的第二个参数



【例 9-3】演示了筛选器的用法。借助于这个页面，可以试验本章中的许多示例，读者可自行练习。虽然这些功能很强大，但是如果不能操作它们的结果，那么它们也就没有什么实际价值。下一节将讨论如何修改匹配集中的项的外观和行为。

【例 9-3】使用 jQuery 筛选器。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 9-3】。

(2) 切换到 Default.aspx 的源视图，在<head>标记中添加如下代码以引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 在<form>标记中添加如下代码：

```
<div>
<h1 title="First Header">一级标题</h1>
<table id="Table1">
  <tr><td>姓名</td><td>电话</td></tr>
  <tr><td>赵艳铎</td><td>01082166054</td></tr>
  <tr><td>葛萌萌</td><td>03172059033</td></tr>
  <tr><td>小石头</td><td>057187982212</td></tr>
  <tr><td>赵飞燕</td><td>02150422312</td></tr>
</table>
<h2>二级  <span style="font-style: italic; font-weight: bold;">标题</span></h2>
<input id="Button1" type="button" value="button" />
<input id="Text1" type="text" />
<br />
<input id="Checkbox1" type="checkbox" />上网
<input id="Checkbox2" type="checkbox" />游戏
<input id="Checkbox3" type="checkbox" />逛街
```



```
<input id="Checkbox4" type="checkbox" />理财
<input id="Button2" type="button" value="全部选中" />
<input id="Button3" type="button" value="全部取消选中" />
</div>
```

(4) 添加“文档就绪函数”，代码如下：

```
<script type="text/javascript">
    $(function () {
        $('#Table1').attr('border', '1');
        $('#Table1').attr('cellpadding', '2');
        $('#Table1').attr('cellspacing', '2');
        $('#Table1 tr:first').css('background-color', 'red');
        $('#Table1 tr:odd').css('background-color', 'green');
        $(':button, :text').css('color', '#ee0033');
        $(':header').css('color', '#800080');
        $(':header:has("span")').css('border-bottom-style', 'dashed');
        $('#Button2').click(function () {
            $(':checkbox').attr('checked', true);
        });
        $('#Button3').click(function () {
            $(':checkbox').attr('checked', false);
        });
    });
</script>
```

(5) 编译并运行程序，在浏览器中打开 Default.aspx 页面，如图 9-4 所示。单击【全部选中】按钮，可以看到复选框全部被选中，如图 9-5 所示。单击【全部取消选中】按钮将取消所有复选框的选中状态。

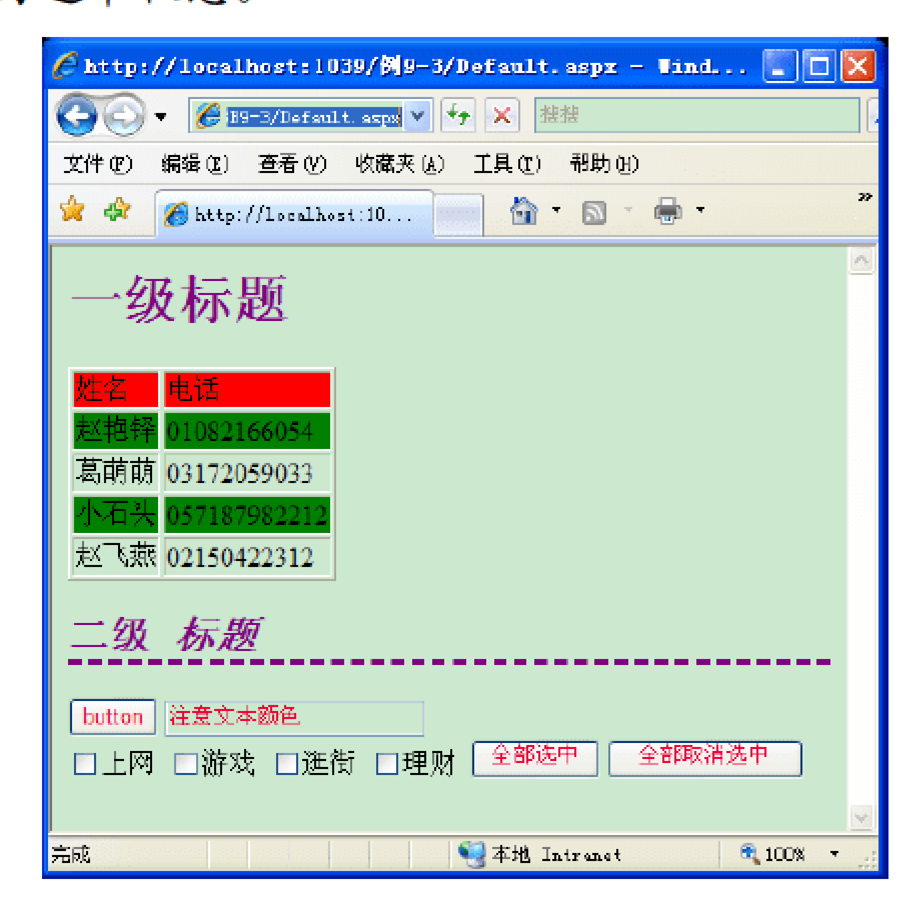


图 9-4 使用筛选器页面效果

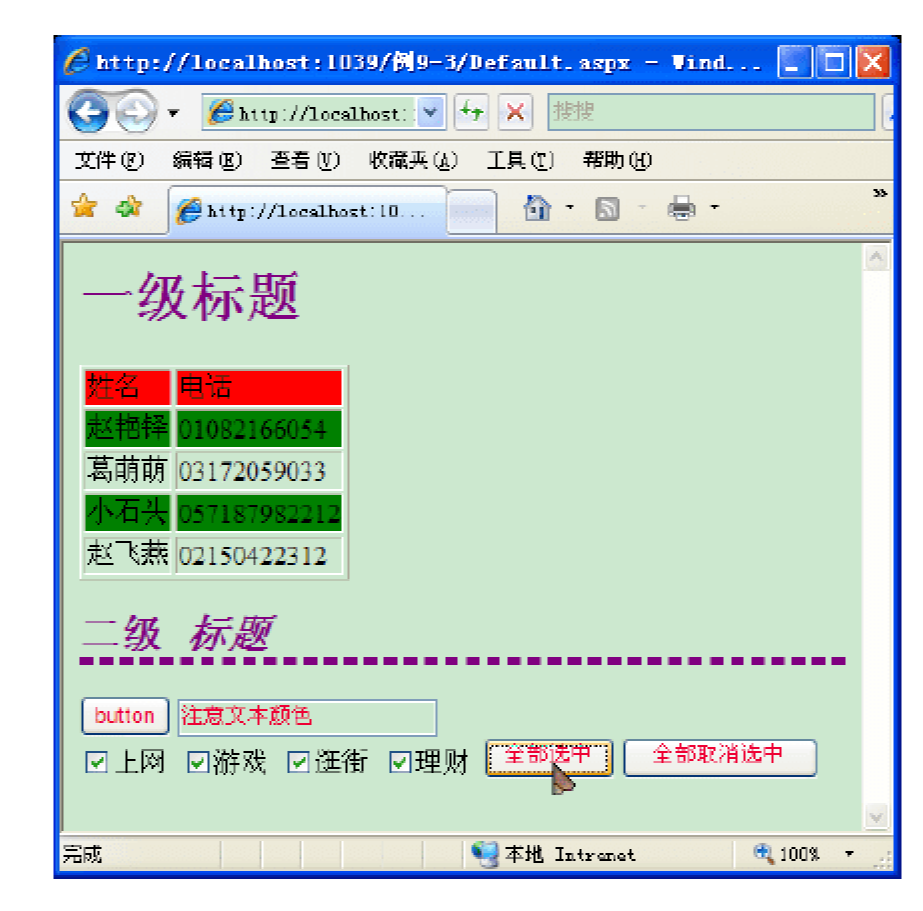


图 9-5 全部选中复选框效果



9.2.4 对匹配集中的项应用 CSS

有了匹配集之后，就需要对它执行一些操作，本节将介绍如何对匹配集中的项应用 CSS 类或者样式。

jQuery 以几种不同的方式支持 CSS。首先，可以使用 css 方法来检索特定的 CSS 值(如某个项的颜色)，以及设置一组元素的一个或多个 CSS 属性。其次，使用 addClass、removeClass、toggleClass 和 hasClass 等方法可以修改或检查对元素应用的 CSS 类。再次，还有以下几种方法可以用来修改元素的尺寸和位置。

- css(name,value)

这个方法用来设置某个匹配元素上的特定的 CSS 属性。name 参数是引用一个 CSS 属性的名称(如 border、color 等)，value 定义了要应用的样式。下面的示例就是修改 h1 元素的背景色：

```
$('h1').css('background-color', 'green');
```


- css(name)

这个方法基于传递给它的属性检索特定的 CSS 值。下面的示例将弹出对话框，内容是二级标题的 span 元素的 font-style 属性值。

```
alert($('h2 span').css('font-style'));
```

可以在 jQuery 脚本中使用这个值；例如，可以用来在斜体和普通字体之间切换 font-style，或者将多个元素设置为相同的类型。

- css(properties)

这是一个功能强大的方法，它可以用来同时设置匹配元素的多个属性。下面的示例将表中所有单元格的颜色修改为红色，内边距设为 10px，字体修改为 Verdana。注意每个属性和属性值直接由冒号(:)分隔，而每个属性/属性值对之间由逗号分隔。完整的属性集包含在一对花括号({})之间：

```
$('#TableId td').css({'color' : 'red', 'font-family' : 'Verdana',
                'padding' : '10px'});
```

- addClass、removeClass 和 toggleClass

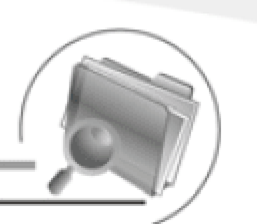
addClass 和 removeClass 方法分别用来在元素中添加和删除类。和普通的 CSS 一样，使用这些方法，比使用 css(properties)方法进行内联 CSS 赋值更好。这样就更容易在一个集中的位置定义 CSS 类，从而使得它们更易于维护和重用。下面的代码将为 h2 元素添加新的 CSS 类：

```
$('h2').addClass('myClass');
```

如果希望再次删除类，则可以调用 removeClass 方法，如下所示：

```
$('h2').removeClass('myClass');
```

如果类还不存在，则 toggleClass 方法将分配一个类，否则将删除类。



这 3 个方法都允许传递多个类，各个类之间用空格分隔。

9.2.5　添加事件处理

事件是许多编程语言中常用的一种技术。在前面的章节中已经介绍了.NET 事件的应用，JavaScript 和 DOM 也不例外，它们在许多地方都提供了事件。例如，许多 HTML 元素(如使用 input type="button"定义的按钮)都有一个 Click 事件，在单击的时候触发。同样地，它们还有 onmouseover 和 onmouseout 事件，当鼠标经过或者离开它们的时候触发。

可以在标记中直接定义事件，也可以将事件处理程序定义为一个函数，例如，下面的两种形式都是合法的：

```
<input type="button" onclick="alert('Hello');" value="提交" />
<input type="button" onclick="SayHello();" value="提交" />
```

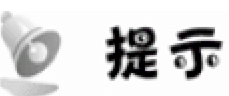


在学习 AJAX 的时候，曾介绍 ASP.NET AJAX 框架的 addHandler 方法，可以在一个独立的代码块中建立处理程序。jQuery 则更进一步，不仅仅允许将事件挂钩到单个元素上，还允许将事件挂钩到整个匹配集上。这种功能极为强大，因为只用几行代码，就可以将处理程序绑定到大量的元素上。例如，为了使表格的外观美观一些，可以设置当鼠标移动到某行时，该行就改变颜色。如果不使用 jQuery，则需要对表的每一行都编写 onmouseover 和 onmouseout 事件。而使用 jQuery，则只需使用如下几行代码：

```
$(function(){
  $('#TableId tr')
       .bind('mouseover', function() { $(this).css('background-color', 'yellow') });
});
```

这些代码将找出#TableId 元素中的全部表行，然后动态分配一个函数，当鼠标悬停在每一行上时，将会调用该函数。要将 onmouseout 绑定到一个新函数，只需对 bind 的第一次调用返回的值再次调用 bind 方法即可。jQuery 方法的优点在于，除了应用某些设计或行为，它们会再次返回匹配集，这样就可以对相同的匹配集调用其他方法。这个概念称为链接(chaining)，在这种概念中，使用一个方法的结果作为另外一个方法的输入，从而产生一个效果链。

```
$('#TableId tr')
    .bind('mouseover', function() { $(this).css('background-color', 'yellow') })
    .bind('mouseout', function() { $(this).css('background-color', '') });
```



上述代码中结束行的分号移动到了最后一行。这样，第二个 bind 方法就绑定到了前一次对 bind 方法的调用上。

上述代码完成三项工作：首先，使用$('#TableId tr')找出表中的全部行。它在返回的匹配集中调用 bind 方法，以便动态挂钩一些行为，当鼠标移动到某一行上时，就会触发这些行为。然后，对第一次调用 bind 方法返回的匹配集再次调用 bind 方法，以便当鼠标从该行移走的时候重设背景色。代码中将颜色设置为一个空字符串(' ')，以便删除 CSS 背景属性，这样就可以看到原来的背景。

在这个示例中，还有一个重点地方需要注意，即设置背景色的方式：

```
$(this).css('background-color', 'yellow')
```

其中，this 关键字指的是应用该项的元素：在本例中就是表行。使用$(this)将得到 jQuery 匹配集(包含单个元素)，可以对其应用常规的 jQuery 方法，如 css 方法，也可以不是以 jQuery 而是对 this 元素执行标准的 JavaScript 方法。例如：

```
this.style.backgroundColor = 'yellow'
```





知识点

在 JavaScript 中，短划线(-)不是一个有效的标识符，所以在 JavaScript 中，所有的短划线都将从属性名中删除。而且，原来紧跟在短划线后面的字母将变为大写方式。所以，CSS 中的 background-color 在 JavaScript 中就变成了 backgroundColor，font-family 就变成了 fontFamily，等等。

绑定事件之后，也可以使用 unbind([type],[fn])方法删除事件绑定，其中第一个参数表示要删除绑定的事件名，第二个参数表示删除的附带参数。例如，下面示例将把刚注册的鼠标移走事件删除掉：

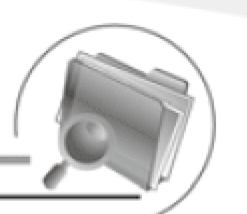
```
$('#TableId tr').unbind('mouseout');
```

另外，bind()方法有一个特例就是 one()方法，它能够匹配元素绑定一个仅能够执行一次的事件处理函数。在每个对象上，这个事件处理函数只会被执行一次。其他规则与 bind()函数相同。这个事件处理函数会接收到一个事件对象，可以通过它来阻止(浏览器)默认的行为。例如：

```
$("p").one("click", function(){
  alert( $(this).text() );
});
```

9.2.6　访问 jQuery 对象

通过选择器或筛选器得到的 jQuery 对象是一个集合，要访问该集合，除了使用索引值以外，还可以使用 jQuery 定义的几个方法和属性。另外，jQuery 还优化并扩展了很多筛选函数，这些函数作为 jQuery 对象的方法直接使用，这样就能够在选择器的基础上更加精确地控制对象。



1. each 方法

each 方法迭代(或循环遍历)一个集合。当需要对匹配集中的项应用某种行为，但是无法使用一个 jQuery 函数完成设置时，就可以使用 each 方法，把希望对每一项执行的函数作为参数传递给 each。例如，下面的 each 示例通过循环遍历匹配集中的每一项，然后调用 alert，将每个单元格的内容显示出来。

```
$('#TableId td').each(function() {
      alert(this.innerHTML);
   });
```

2. size()和 length

size()方法能够返回 jQuery 对象中元素的个数，而 length 属性与 size()方法功能相同。例如，下面的代码使用 size()方法和 length 属性返回值都为 2。

```
<span>文本块 1</span>
<span>文本块 2</span>
<script language="javascript" type="text/javascript">
      alert($("span").size()); //返回值为 2
      alert($("span").length); //返回值为 2
</script>
```



3. get 方法

get()方法能够把 jQuery 对象转换为 DOM 中的元素集合。例如，在下面示例中，使用$()函数获取所有 span 元素，然后用 get()方法把该 jQuery 对象转换为 DOM 集合，再调用 JavaScript 数组方法 reverse()把数组元素的位置颠倒过来。最后为数组中第一个元素设置字体为红色，最终效果是文本“文本块 2”显示为红色。

```
<span>文本块 1</span><span>文本块 2</span>
<script language="javascript" type="text/javascript">
var spans = $("span").get().reverse(); //把当前 jQuery 对象转换为 DOM 对象并颠倒它们的位置
spans[0].style.color = "red"; //把当前 jQuery  对象设置为红色
</script>
```

也可以使用 get(index)方法获取指定索引值的元素对象。

4. index 方法

Index 方法用于获取 jQuery 对象中指定元素的索引值，如果找到了匹配的元素，从 0 开始返回；如果没有找到匹配的元素，则返回－1。

如果不给 index()方法传递参数，那么返回值就是这个 jQuery 对象集合中第一个元素相对于其同辈元素的位置；如果参数是一组 DOM 元素或者 jQuery 对象，那么返回值就是传递的元素相

对于原先集合的位置；如果参数是一个选择器，那么返回值就是原先元素相对于选择器匹配元素中的位置。

例如，在下面这个示例中，所有的调用都返回 1。

```
<ul>
  <li id="foo">foo</li>
  <li id="bar">bar</li>
  <li id="baz">baz</li>
</ul>
<script language="javascript" type="text/javascript">
$('li').index(document.getElementById('bar'));   //1，返回这个对象在原先集合中的索引位置
$('li').index($('#bar'));   //1，传递一个 jQuery 对象
$('li').index($('li:gt(0)'));   //1，传递一组 jQuery 对象，返回这个对象中第一个元素在原先集合中的索引位置
$('#bar').index('li');   //1，传递一个选择器，返回#bar 在所有 li 中的做引位置
$('#bar').index();   //1，不传递参数，返回这个元素在同辈中的索引位置。
</script>
```

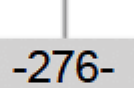


5. 筛选函数

jQuery 定义了很多能够从选取对象中过滤部分元素的方法，这些方法是对选择器功能的补充。如表 9-3 所示列出了一些常用的筛选函数。

表 9-3 筛选函数

筛 选 函 数	说　　明
eq(index)	获取指定索引值位置上的元素，索引值从 0 开始
hasClass(class)	检查当前元素是否含有某个特定的类，如果有，则返回 true
filter(expr)	筛选与指定表达式匹配的元素集合。该方法用于缩小匹配范围，用逗号分隔多个表达式
filter(fn)	筛选出与指定函数返回值匹配的元素集合
is(expr)	用一个表达式来检查当前选择的元素集合，如果其中至少有一个元素符合给定的表达式就返回 true
map(callback)	将一组元素转换成其他数组(不论是否是元素数组)
not(expr)	删除与指定表达式匹配的元素
slice(start,[end])	选取一个匹配的子集，与原来的 slice 方法类似
add(expr)	把与表达式匹配的元素添加到 jQuery 对象中。这个函数可以用于连接分别与两个表达式匹配的元素结果集
children([expr])	取得一个包含匹配的元素集合中每一个元素的所有子元素的元素集合
contents()	查找匹配元素内部所有的子节点(包括文本节点)。如果元素是 iframe，则查找文档内容
find(expr)	搜索所有与指定表达式匹配的元素。这个函数是找出正在处理的元素的后代元素
next([expr])	取得一个包含匹配的元素集合中每一个元素紧邻的后面同辈元素的元素集合



(续表)

筛选函数	说明
nextAll([expr])	查找当前元素之后的所有元素
parent([expr])	取得一个包含着所有匹配元素的唯一父元素的元素集合
parents([expr])	取得一个包含着所有匹配元素的祖先元素的元素集合(不包含根元素)
prev([expr])	取得一个包含匹配的元素集合中每一个元素紧邻的前一个同辈元素的元素集合
prevAll([expr])	查找当前元素之前所有的同辈元素，可以用表达式过滤
siblings([expr])	取得一个包含匹配的元素集合中每一个元素的所有唯一同辈元素的元素集合。可以用可选的表达式进行筛选
andSelf()	加入先前所选的当前元素中，对于筛选或查找后的元素，要加入先前所选元素时将会很有用
end()	回到最近的一个“破坏性”操作之前，即将匹配的元素列表变为前一次的状态

例如，有如下 HTML 代码：

```
<p><span>Hello</span>, how are you?</p>
```

那么，$("p").find("span")的结果为：Hello。

9.2.7 使用 jQuery 的效果

在【例 9-2】中用了 slideUp 和 slideDown 来逐渐隐藏和显示元素，但是这只是 jQuery 提供的诸多效果和动画方法中的两种方法。本节将介绍其他一项常用的方法，如表 9-4 所示。

表 9-4 常用的动画效果方法

方法	用途
show() hide()	通过递减 height、width 和 opacity(使它们变为透明)隐藏或者显示匹配元素。两种方法都允许定义固定的速度(慢、中、快)或者一个定义动画持续时间(单位为毫秒)的数字。示例如下： $('h1').hide(1000); $('h1').show(1000);
toggle()	toggle 方法在内部使用 show 和 hide 来改变匹配元素的显示方式。即，可见元素将被隐藏，不可见元素将会显示。示例如下： $('h1').toggle(2000);
slideDown() slideUp(() slideToggle()	类似于 hide 和 show，这些方法隐藏或显示匹配元素。但是，这是通过将元素的 height 从当前尺寸调整为 0，或者从 0 调整为初始尺寸来实现的。slideToggle 方法会展开隐藏的元素，卷起可见的元素，从而可以使用一个动作重复地显示和隐藏元素。示例如下： $('h1').slideUp(1000).slideDown(1000); $('h1').slideToggle(1000);



(续表)

方　法	用　途
fadeIn() fadeOut() fadeTo()	这些方法通过修改匹配元素的不透明度显示或隐藏它们。fadeOut 将不透明度设置为 0，使元素完全透明，然后将 CSS display 属性设置为 none，从而完全隐藏元素。fadeTo 允许指定一个不透明度(0 到 1 之间的一个数字)，以便决定元素的透明程度。全部 3 个方法都允许定义一个固定速度(慢、中、快)，或者一个定义了动画持续时间(单位为毫秒)的数字。示例如下： $('h1').fadeOut(1000); $('h1').fadeIn(1000); $('h1').fadeTo(1000, 0.5);
animate()	在内部，animate 用于许多动画方法，例如 show 和 hide。但是，也可以在外部使用它，从而可以更灵活地以动画方式显示匹配元素。例如下面这个示例： $('h1').animate({ 　　　opacity: 0.4, 　　　marginLeft: '50px', 　　　fontSize: '50px' 　　}, 1500); 这段代码接受一个 h1 元素，将其字体大小设置为 50 像素，将其不透明度设置为 0.4 以使元素半透明，并将其左页边距设置为 50 像素，从而在 1.5 秒的时间内平滑地进行动画显示。animate 方法的第一个参数是一个对象，它保存一个或者多个想要动画显示的属性，每个属性之间以逗号分隔。注意，需要使用 JavaScript 的 marginLeft 和 fontSize，而不是 CSS margin-left 和 font-size 属性。只能动画显示接受数值的属性。也就是说，可以使用 margin、fontSize、opacity 等属性，但是不能使用 color 或 fontFamily 这样的属性
stop()	停止所有在指定元素上正在运行的动画，如果队列中有等待执行的动画(并且第一个参数不是 true)，则将被马上执行
delay()	设置一个延时来推迟执行队列中之后的项目，用于将队列中的函数延时执行。该方法既可以推迟动画队列的执行，也可以用于自定义队列。例如下面的代码将在.slideUp()和.fadeIn()之间延时 1 秒： $('h1').slideUp(1000).delay(1000).fadeIn(1000);

下面来看一个 jQuery 实现的动画效果。

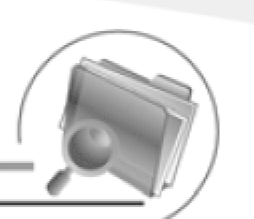
【例 9-4】使用 jQuery 的效果动态显示图片。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 9-4】。

(2) 切换到 Default.aspx 的源视图，在<head>标记中添加如下代码以引入 jQuery 库：

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(3) 新建一个文件夹 images，并在其中添加 5 个图片文件 p1.jpg~p5.jpg。



(4) 在<form>标记中添加如下代码:

```
<div id="photoShow">
    <div class="photo">
        <img src="images/p1.jpg" />
        <span>蓝天与山脉</span>
    </div>
    <div class="photo">
        <img src="images/p2.jpg" />
        <span>夕阳无限好</span>
    </div>
    <div class="photo">
        <img src="images/p3.jpg" />
        <span>小荷才露尖尖角</span>
    </div>
    <div class="photo">
        <img src="images/p4.jpg" />
        <span>寒冷的冬天</span>
    </div>
    <div class="photo">
        <img src="images/p5.jpg" />
        <span>清华大学</span>
    </div>
</div>
```



(5) 上述代码用到了一些 CSS 样式，所以需要添加这些样式的定义，本例直接在 Default.aspx 中定义内嵌式样式，首先在<head>标记中添加如下代码:

```
<style type="text/css">
    #photoShow{
        border: solid 1px #C5E88E;
        overflow: hidden; /*图片超出 DIV 的部分不显示*/
        width: 490px;
        height: 169px;
        background: #C5E88E;
        position: absolute;
    }
    .photo{
        position: absolute;
        top: 0px;
        width: 400px;
```

```
            height: 169px;
        }
        .photo img{
            width: 400px;
            height: 169px;
        }
        .photo span{
            padding: 5px 0px 0px 5px;
            width: 400px;
            height: 30px;
            position: absolute;
            left: 0px;
            bottom: -32px; /*介绍内容开始的时候不显示*/
            background: black;
            color: #FFFFFF;
        }
    </style>
```

(6) 添加“文档就绪函数”，代码如下：

```
    <script type="text/javascript">
        $(document).ready(function () {
            var imgDivs = $("#photoShow>div");
            var imgNums = imgDivs.length; //图片数量
            var divWidth = parseInt($("#photoShow").css("width")); //显示宽度
            var imgWidth = parseInt($(".photo>img").css("width")); //图片宽度
            var minWidth = (divWidth - imgWidth) / (imgNums - 1); //显示其中一张图片时其他图片的显示
宽度
            var spanHeight = parseInt($("#photoShow>.photo:first>span").css("height")); //图片介绍信息的高度
            imgDivs.each(function (i) {
                $(imgDivs[i]).css({ "z-index": i, "left": i * (divWidth / imgNums) });
                $(imgDivs[i]).hover(function () {
                    $(this).find("span").stop().animate({ bottom: 0 }, "slow");
                    imgDivs.each(function (j) {
                        if (j <= i) {
                            $(imgDivs[j]).stop().animate({ left: j * minWidth }, "slow");
                        } else {
                            $(imgDivs[j]).stop().animate({ left: (j - 1) * minWidth + imgWidth }, "slow");
                        }
                    });
                }, function () {
```



```
                imgDivs.each(function (k) {
                    $(this).find("span").stop().animate({ bottom: -spanHeight }, "slow");
                    $(imgDivs[k]).stop().animate({ left: k * (divWidth / imgNums) }, "slow");
                });
            });
        });
    });
</script>
```

在上述代码中，首先定义了一些变量，然后使用 each()函数在每一个匹配的元素进行事件处理。通过 hover() 函数来处理鼠标的 hover 事件。在这里所有的动画效果都是通过 animate()函数修改 CSS 来控制元素的显示位置来实现的。调用 animate()函数前调用 stop()函数是用来停止当前元素的所有执行中的事件。

(7) 编译并运行程序，在浏览器中打开 Default.aspx 页面，初始状态如图 9-6 所示。当鼠标进入图片区域后将以大图显示当前所在的图片，相应的其他图片将缩小，如图 9-7 所示。

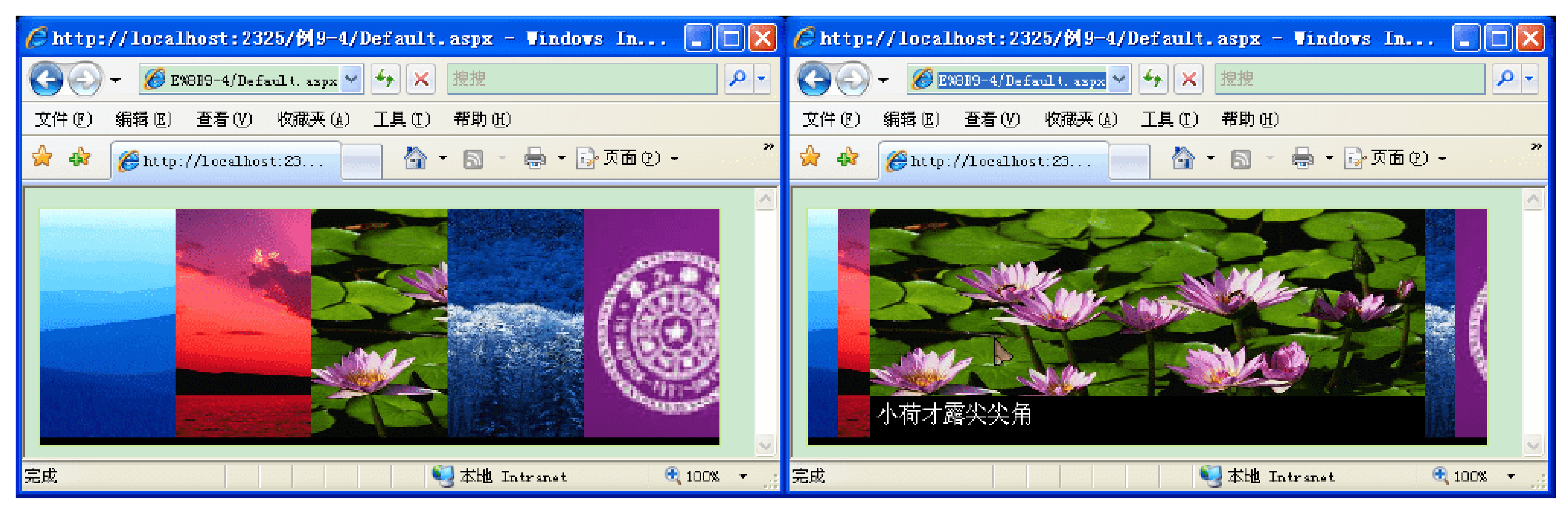


图 9-6　平均大小显示所有图片　　　　图 9-7　鼠标进入图片区域动态显示

虽然一些代码看上去十分复杂，但是使用 jQuery 实现一些特殊效果仍然相对简单。

9.3　jQuery 扩展应用

相信通过前面的学习，读者已经看到了 jQuery 的强大和优势，但是 jQuery 的功能远不止这些。jQuery 提供了一个灵活的插件体系结构，使得插件开发者编写的功能可以很容易地通过包含一个或多个 JavaScript 文件以及调用一个或者多个方法进行重用。jQuery 社区积极地开发出了数百种实用的插件，可以毫不费力地把它们添加到页面中。

jQuery 插件非常流行，以至于 jQuery 站点专门用了一部分空间来提供插件：http://plugins.jquery.com/。除了 jQuery.com 上的插件存储库以外，Internet 上还有其他许多插件。

通常，使用 jQuery 插件的步骤如下：

(1) 在 Internet 上找到并下载要使用的插件。

(2) 在项目中包含插件，即将下载的.js 文件添加到 Scripts 文件夹中。

(3) 在页面中添加对插件文件的引用。如果需要大量使用该插件，则把它添加到母版页中。

(4) 通过编写代码使用插件。具体的代码取决于使用的插件。一般可以在线找到插件的示例或 readme 文件。

9.3.1 使用 jQuery 插件

从 jQuery 网站或 Internet 网站下载的插件通常都有示例程序，使用这些插件都十分简单，开发人员不必关心它的工作原理，只需参照示例使用插件、执行代码即可实现相应的效果。

下面将介绍水印效果的 jQuery 插件的使用，可以在 http://plugins.jquery.com/project/updnWatermark 上找到该插件。

【例 9-5】使用水印效果的 jQuery 插件。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 9-5】。

(2) 将下载的 jQuery 插件解压出来，将得到的 jquery.updnWatermark.js 文件添加到项目的 Scripts 文件夹中，将 Sample.css 文件添加到 Styles 文件夹中。

(3) 切换到 Default.aspx 的源视图，在<head>标记中添加如下代码引入 jQuery 库和 CSS 样式文件:

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
<script src="Scripts/jquery.updnWatermark.js" type="text/javascript"></script>
<link type="text/css" rel="Stylesheet" href="Styles/Sample.css" />
```

(4) 添加一个 TextBox 控件到页面中，要想使插件将水印文本添加到文本框中，需要设置想添加水印的每个控件的 ToolTip 属性。标记中的 ToolTip 属性将映射到最终 HTML 中的 title 属性。设置 TextBox 控件的 ToolTip 属性，添加一些提示性文本，此时<form>标记中的代码如下:

```
<div>
    <h2>updnWatermark 使用示例</h2>
    姓名：<asp:TextBox ID="TextBox1" runat="server" ToolTip="输入姓名，如 葛萌萌"></asp:TextBox>
</div>
```

(5) 接下来就是在文档就绪函数中调用插件的主方法，设置控件的水印效果。当输入$.时将弹出智能提示列表，在该列表中可以看到 updnWatermark，如图 9-8 所示。

(6) 可以参照下载的插件的示例代码，添加如下代码:

```
<script type="text/javascript">
    $(function () {
        $.updnWatermark.attachAll();
    });
</script>
```

提示

不需要选择任何项，只需要调用 jQuery 对象的 updnWatermark 方法(使用$快捷方式)，而不必指定任何选择器。然后 updnWatermark 方法将扫描表单，查找具有 title 属性的表单字段。也可以选择传入一个定义了文本标记的 CSS 类。

(7) 编译并运行程序，运行效果如图 9-9 所示。将光标聚焦到文本框控件上时，文本将会消失。不在控件中输入值并按失去焦点时，文本将会重新显示。

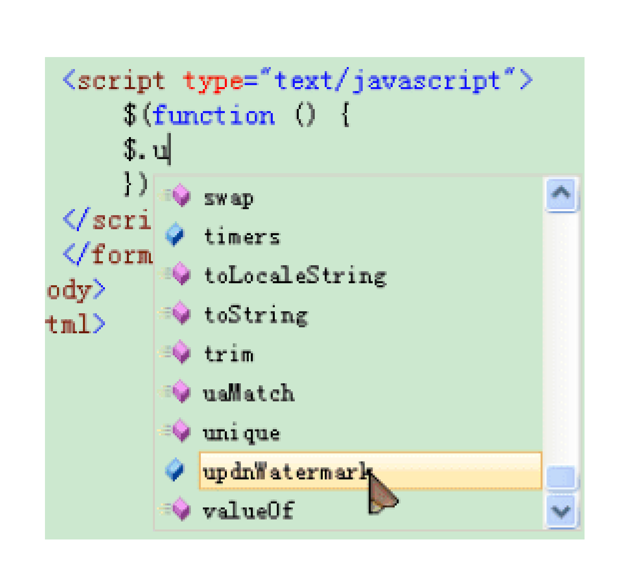


图 9-8　代码的智能提示

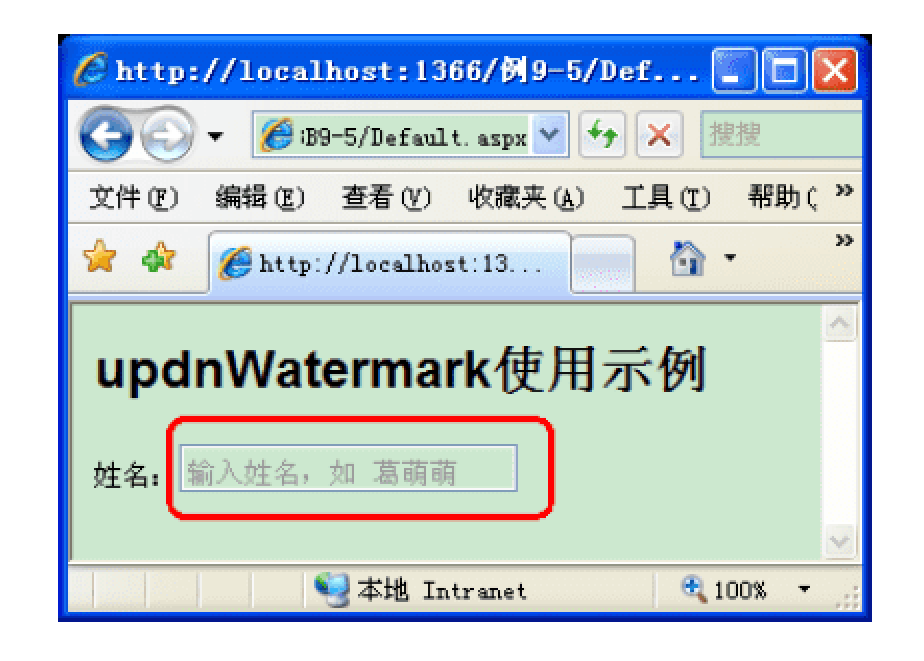


图 9-9　水印效果



知识点

这个水印插件很智能，它并不会把文本放到文本框中。相反，它会快速创建一个标记，并将其放到文本框上，从而使文本看上去包含在文本框中。这个解决方案的智能之处在于，因为实际的字段没有改变，所以控件的验证仍然与以前一样。验证控件会看到一个空文本框，如果没有输入有效数据，但是却想提交表单，验证控件就会触发一个验证弹出窗口。

9.3.2　编写 jQuery 插件

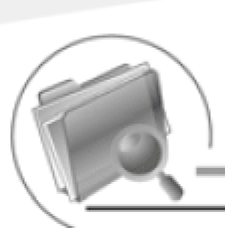
本节将通过一个简单的例子来介绍如何编写 jQuery 插件，要编写的这个 jQuery 插件很简单，实现的功能是：给指定表格加上鼠标所在行高亮显示。

【例 9-6】编写一个给指定表格加上鼠标指针所在行高亮显示的 jQuery 插件，并提供测试页面。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【例 9-6】。

(2) 在 Scripts 文件夹中通过【添加新项】命令添加一个 JScript 文件 jQuery.tabHighLight.js。

(3) 在新建的文件中添加如下代码：




```
(function ($) {
    $.fn.lightRow = function (row_color) {
        var default_color = "#669900";    //默认颜色
        row_color = (row_color === undefined) ? default_color : row_color;
        $(this).find("tr").each(function () {
            $(this).mouseover(function () {
                $(this).attr("old_color", $(this).css("background-color"));   //创建属性保存旧颜色
                $(this).css("background-color", row_color);   //使用新颜色
            }).mouseout(function () {
                $(this).css("background-color", $(this).attr("old_color"));    //恢复旧颜色
                $(this).removeAttr("old_color");   //删除保存旧颜色的属性
            });
        });
        return $(this);   //保持操作链
    }
})(jQuery);
```

上述代码的核心部分与【例 9-3】中添加的类似，都是添加了鼠标进入和移出事件，所不同的是头和尾的格式，头部定义了方法的调用方式(方法名是 lightRow，并且可以有一个可选参数指定颜色)，尾部返回原调用对象，并在末尾用(jQuery)标识这是一个 jQuery 插件。

(4) 在 Default.aspx 页面中添加一个表格，代码如下：

```
<table id="table1">
    <tr>
        <td>姓名</td><td>电话</td>
    </tr>
    <tr>
        <td>赵艳铎</td><td>15910801234</td>
    </tr>
    <tr>
        <td>葛萌萌</td><td>13831705678</td>
    </tr>
    <tr>
        <td>小石头</td><td>01082166054</td>
    </tr>
    <tr>
        <td>金百合</td><td>03172059876</td>
    </tr>
</table>
```

(5) 在<head>标记中添加如下代码引入 jQuery 库：



```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
<script src="Scripts/jQuery.tabHighLight.js" type="text/javascript"></script>
```

(6) 接下来就是在文档就绪函数中调用插件的方法，添加如下代码：

```
<script type="text/javascript">
    $(function () {
        $('#table1').attr({border: '1', cellpadding: '2',cellspacing: '2'});
        $('#table1').lightRow('red');
    });
</script>
```

(7) 编译并运行程序，运行效果如图 9-10 所示。

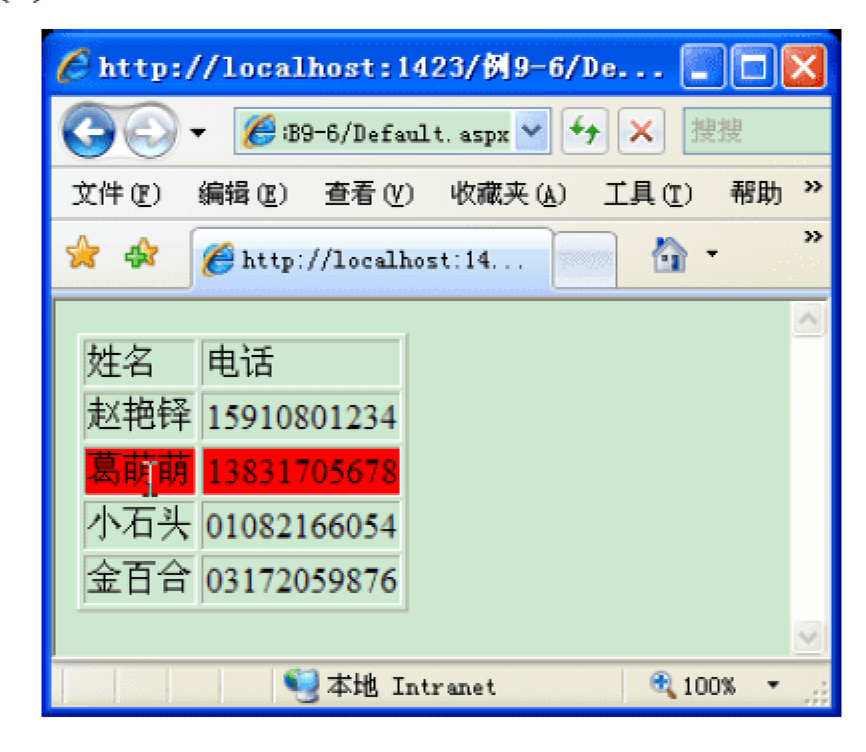


图 9-10 测试所编写的 jQuery 插件



知识点

在输入$('#table1').后弹出的智能提示列表中将可以看到所定义的 lightRow。



9.4 上机练习

本章的上机练习将介绍 jQuery 对 AJAX 的支持，jQuery 封装了 XMLHttpRequest 组件并初始化，还封装了 AJAX 请求中的各种基本操作，并把这些操作定义为简单的方法。另外，它把 AJAX 请求中各种状态封装为事件，这样只要调用对应的事件就可以快速执行绑定的回调函数。

(1) 启动 VWD 2010，选择【文件】|【新建网站】命令，新建网站【上机练习 9】。

(2) 新建一个 HTML 文件 test.htm，并在其中输入如下代码：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="ContentType"content="text/html; charset=utf8"/>
<title>Ajax 请求</title>
</head>
<body>
<li>第一章 快速入门</li>
```

```
<li>第二章  Web 服务</li>
<li>第三章  Ajax</li>
</body>
</html>
```

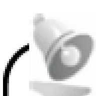
(3) 在 Default.aspx 页面的<head>标记中添加如下代码以引入 jQuery 库:

```
<script src="Scripts/jquery-1.4.1.min.js" type="text/javascript"></script>
```

(4) 在 Default.aspx 页面中添加一个
标记对，然后添加如下代码:

```
<script type="text/javascript">
    $('ul').load("test.htm");
</script>
```

load()方法能够载入远程 HTML 文件并插入匹配元素中。默认使用 GET 方式，传递附加参数时自动转换为 POST 方式。jQuery 会自动从 text.htm 文档中提取 body 元素包含的代码，并把这些代码插入匹配的 ul 元素中。



提示

在使用 load()方法时，所有页面的字符编码应该设置为 utf8，否则 jQuery 在加载文档时会显示乱码。另外，匹配的元素应该只有一个，否则系统会出现异常。

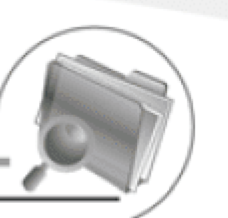
(5) 为了演示其他的 AJAX 方法，在页面中继续添加其他元素。在<ul>标记的下面添加两个 Input(text)控件和两个 Input(button)控件。代码如下:

```
姓名：<input id="Text1" type="text" /><br />
城市：<input id="Text2" type="text" /><br />
<input id="Button1" type="button" value="Ajax Get" />
<input id="Button2" type="button" value="Ajax" />
```

(6) 添加两个 Web 窗体，名称分别为 get.aspx 和 post.aspx。

(7) 为两个按钮绑定事件处理程序，添加如下 jQuery 代码:

```
$('#Button1').bind('click', function () {
    $.get("get.aspx", { user: $('#Text1')[0].value, city: $('#Text2')[0].value },
        function (msg) {
            alert(msg);
        }
    );
});
$('#Button2').bind('click', function () {
    var str = "user="+ $('#Text1')[0].value+"&city="+ $('#Text2')[0].value;
```



```
        $.ajax({ url: "post.aspx",
            type: "POST",
            data: str,
            success: function (msg) {
                alert(msg);
            }
        });
});
```

在上面 Button1 的单击事件代码中，使用 jQuery 对象的 get()方法向 get.aspx 文件发送一个请求，并以 GET 方式向服务器传递两个参数，服务器响应之后会把返回值存储在回调函数参数中，弹出消息提示框；Button2 的单击事件中，使用 AJAX 方法向 post.aspx 文件发送请求，并以 POST 方式传递参数，最后调用回调函数获取响应信息。虽然是调用不同的方法，但实现的功能都是相同的。

(8) 在 get.aspx 页面中，删除 Page 指令之外的其他代码，然后添加如下代码：

```
<%
    string str= Request.QueryString["user"]+ "是" + Request.QueryString["city"] + "人";
    Response.AddHeader("ContentType", "text/html;charset=utf8");
    Response.Write(str);
%>
```

(9) 类似地，在 post.aspx 页面中，也删除 Page 指令之外的代码，添加下面的代码：

```
<%
    string str= Request.Form["user"]+ "是" + Request.Form["city"] + "人";
    Response.AddHeader("ContentType", "text/html;charset=utf8");
    Response.Write(str);
%>
```

(10) 编译并运行程序，在浏览器中打开 Default.aspx 页面，如图 9-11 所示，在文本框中输入姓名和城市后，单击下面的 AJAX 按钮，将弹出提示对话框，如图 9-12 所示。





图 9-11　页面运行效果

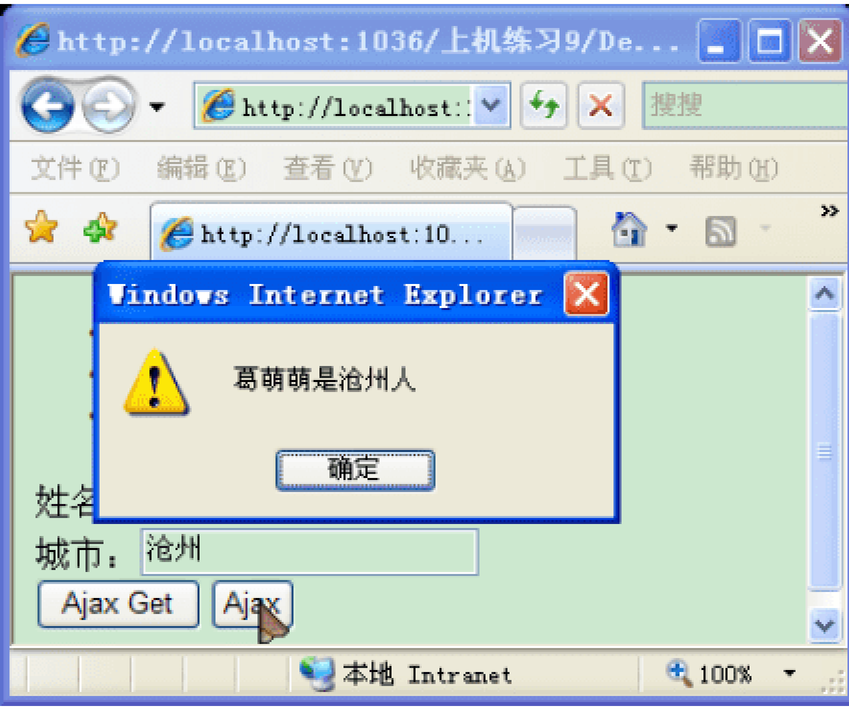
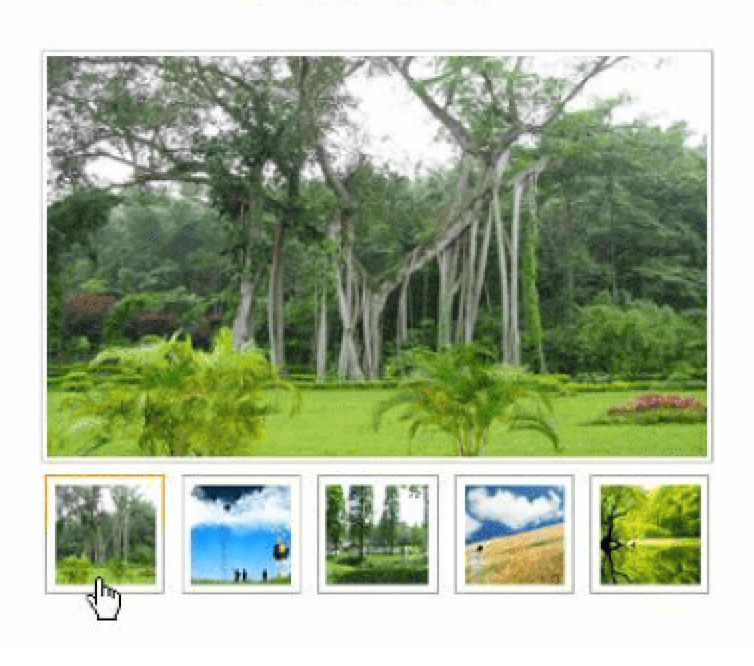


图 9-12　AJAX 调用结果显示

9.5 习题

1. 在 jQuery 中，使用什么作为在页面中查找元素的快捷方式？
2. 所有的 jQuery 选择器都返回什么？
3. jQuery 的层级选择器有哪些？分别有什么功能。
4. bind()方法和 one()方法有什么不同之处？
5. size()和 length()有何异同点？
6. slideUp 和 slideDown 有什么区别？两种方法都接受什么重要参数？
7. jQuery 方法在应用某些设计或行为以后，返回什么？
8. jQuery.get()方法与 jQuery.post()方法有何区别？
9. 在使用 load()方法时，所有页面的字符编码应该设置为什么方式？
10. 上机练习，使用 jQuery 实现如下效果：

- 图片画廊，效果如图 9-13 所示，当用户移动鼠标指针到缩微图上时，该图会自动被放大显示在上面画框内。



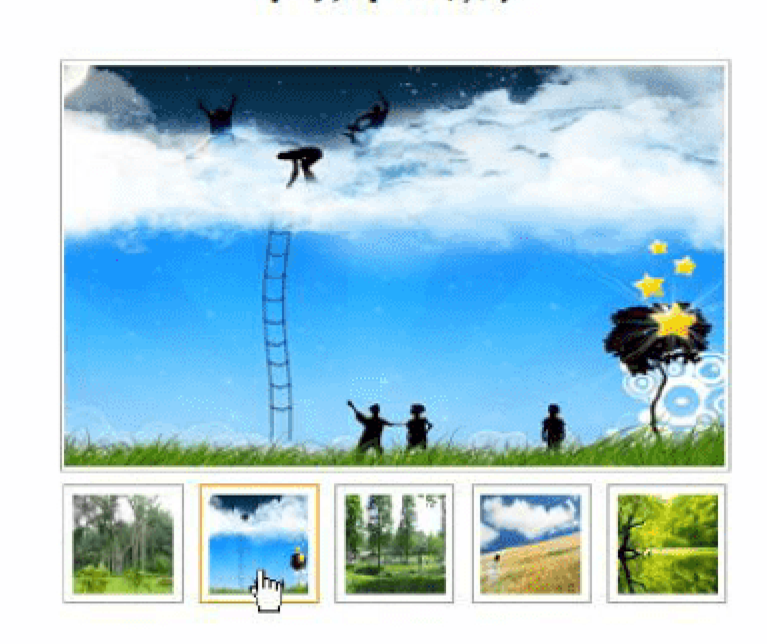


图 9-13　图片画廊效果

- 动态调整区域大小，效果如图 9-14 所示，当用户单击【开始动画】链接时将自动调整区域的大小。

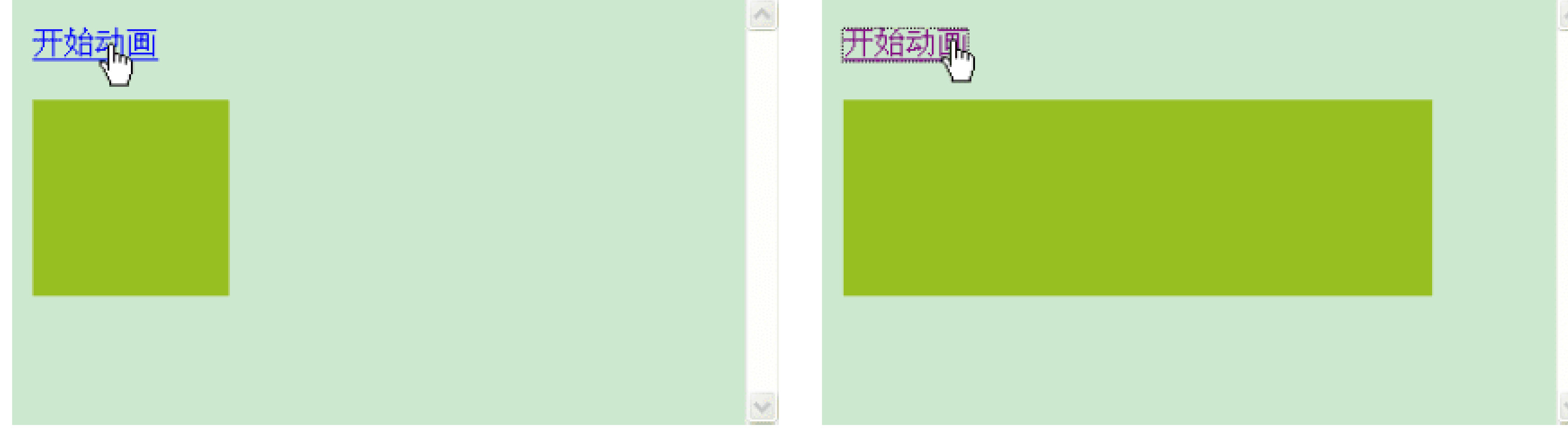
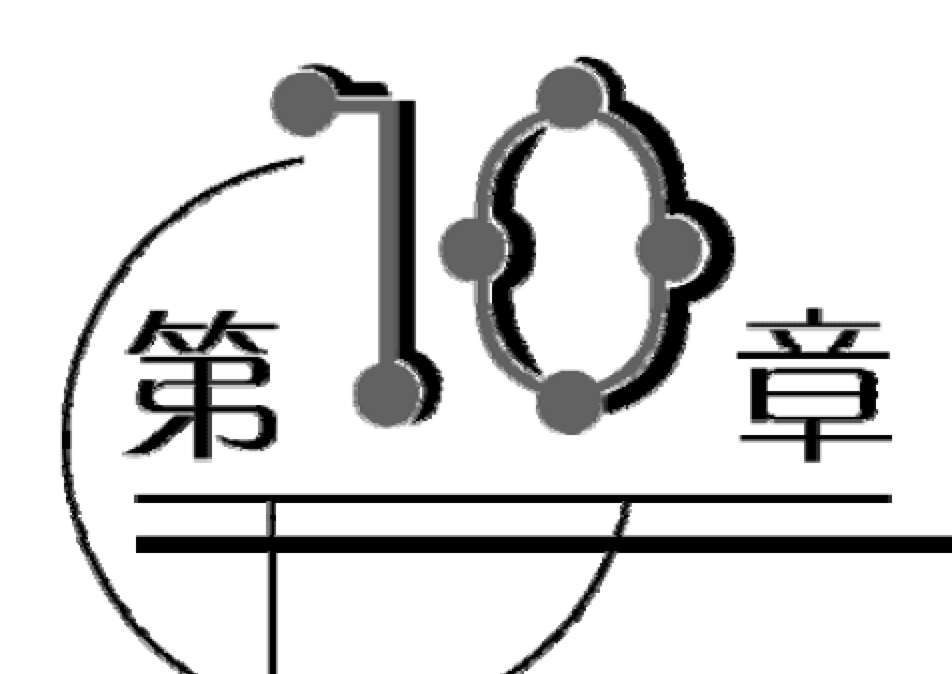


图 9-14　自动调整大小动画效果



第10章 部署 Web 站点

学习目标

本章主要介绍 Web 应用程序的部署，包括复制 Web 站点、在 IIS 下运行站点和将数据库移动到远程服务器。为了让 Internet 上的用户能够访问 Web 站点，需要将它发布到与 Internet 相连的生产服务器上。通过本章的学习，应重点掌握 Web 站点的部署方法。

本章重点

- 复制 Web 站点
- 在 IIS 下运行 Web 站点
- 将数据移到远程服务器

10.1 部署 Web 站点

网站或 Web 应用程序设计开发完成后，需要发布才能让用户访问。使用什么类型的服务器以及将它定位在哪里的服务器，这取决于具体的需求和预算。可以将站点驻留在阁楼里具有私有 Internet 连接的家用服务器上，或者使用能够直接连接到 Internet 主干的外部(通常是商业)方提供商的服务器来驻留它。VWD 2010 提供了发布网站的功能，该功能将网站编译为一组可以通过 IIS 直接执行的文件，然后将这些文件复制到目标 Web 服务器上。

10.1.1 部署前的准备工作

当在开发环境中实现 Web 站点的第一个版本时，管理站点及其源代码就非常简单。只有站点源代码的一个版本，因此维护非常容易。然而，一旦将站点投入到生产环境中，就拥有站点的

两个版本：一个在生产环境中运行，另一个用于开发。这就很难保持同步。例如，在生产环境中可能使用不同的数据库和连接字符串。如果激活站点时在代码中直接进行所有修改，那么在下一次更新过程中很可能会重写某些设置，从而产生不必要不希望看到的结果。

怎么管理相同 Web 站点的不同版本呢？简单的方法就是将某些硬编码的设置移动到 Web.config 文件中。Web.config 配置文件使得在开发环境和生产环境中进行不同设置变得很容易。本书前面已经多次使用 Web.config 文件来存储与连接字符串有关的信息，还介绍了在 <appSettings>元素中通过<add>标记添加用户自定义变量。例如：

```
<appSettings>
<add key="CopyRight" value="版权所有小石头网站" />
</appSettings>
```

在程序中可以使用 System.Web.Configuration 命名空间的 WebConfigurationManager 类来获取，从而只使用一行代码就能从这些部分检索数据。

另外，还可以使用表达式语法来访问<appSettings>元素中的数据，格式如下：

```
<%$ AppSettings:AppSettingKeyName %>
```

其中，AppSettingKeyName 表示在 Web.config 文件中定义的键。例如，要在页面上添加一个 Literal 控件，用来显示网站的版权通告，则可以这样设置 Literal 控件的 Text 属性：

```
<asp:Literal ID="Copyright" runat="server" Text="<%$ AppSettings:CopyRight %>" />
```

通常将生产环境和开发环境中参数不一样的值保存在 Web.config 文件中，通过上述方法访问这些变量的值，从而使站点变得更容易部署和维护。将硬编码的应用程序设置移动到 Web.config 文件之后，部署过程的下一步就是创建 Web 站点的副本。

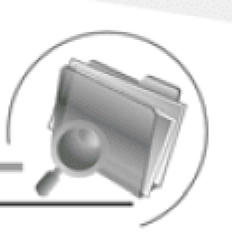
10.1.2 复制 Web 站点

在开发站点的过程中，使用 VWD 配置的内置 Web 服务器。虽然这个服务器对于本地开发非常好，但在生产环境中就不能使用它。要将站点投入到生产环境中使用，需要将它部署到运行 IIS 的计算机中——IIS(Internet Information Services)是 Microsoft 的专业 Web 服务器。

要将站点部署到生产服务器中，可以使用如表 10-1 所示的部署目标(来自 VWD 内部)。

表 10-1　部署目标

部 署 选 项	描　　述
文件系统	该选项允许在开发计算机或网络化计算机的本地文件系统上创建站点副本。如果稍后要将这些文件手动移动到生产服务器中，那么这个选项就很有用
本地 IIS	该选项允许创建将在本地 IIS 安装下运行的站点的副本
FTP 站点	该选项允许使用 FTP 将组成 Web 应用程序的文件发送到远程服务器中



(续表)

部署选项	描　述
远程站点	该选项允许将组成 Web 应用程序的文件发送到远程 IIS 服务器中。要使这个选项生效，远程服务器需要安装 Front Page Server Extensions。查看 IIS 附带的文档或者咨询远程服务器的管理员可以获得使用这个选项的帮助

如果使用的是 Visual Studio 的商业版本，可以使用 VWD 提供的两种主要的部署方法来访问这 4 个部署选项：复制网站和发布网站。如果使用的是免费的 Express 版本，则只能使用复制网站。本书只介绍复制网站。

复制网站命令可以使用 4 个传输选项中的任何一项来创建站点的副本。这是将站点快速复制到其他位置的好方法。

【例 10-1】使用网站复制命令将网站复制到本地 IIS。

(1) 启动 VWD 2010，选择【文件】|【打开网站】命令，打开上一章的网站【例 9-6】。

(2) 本例中没有需要移到 Web.config 文件中的参数信息，所以不用做任何修改。选择【网站】|【复制网站】命令，打开【复制网站】对话框，如图 10-1 所示。



知识点

也可以在【解决方案资源管理器】面板的工具栏中单击复制网站图标，打开【复制网站】对话框。

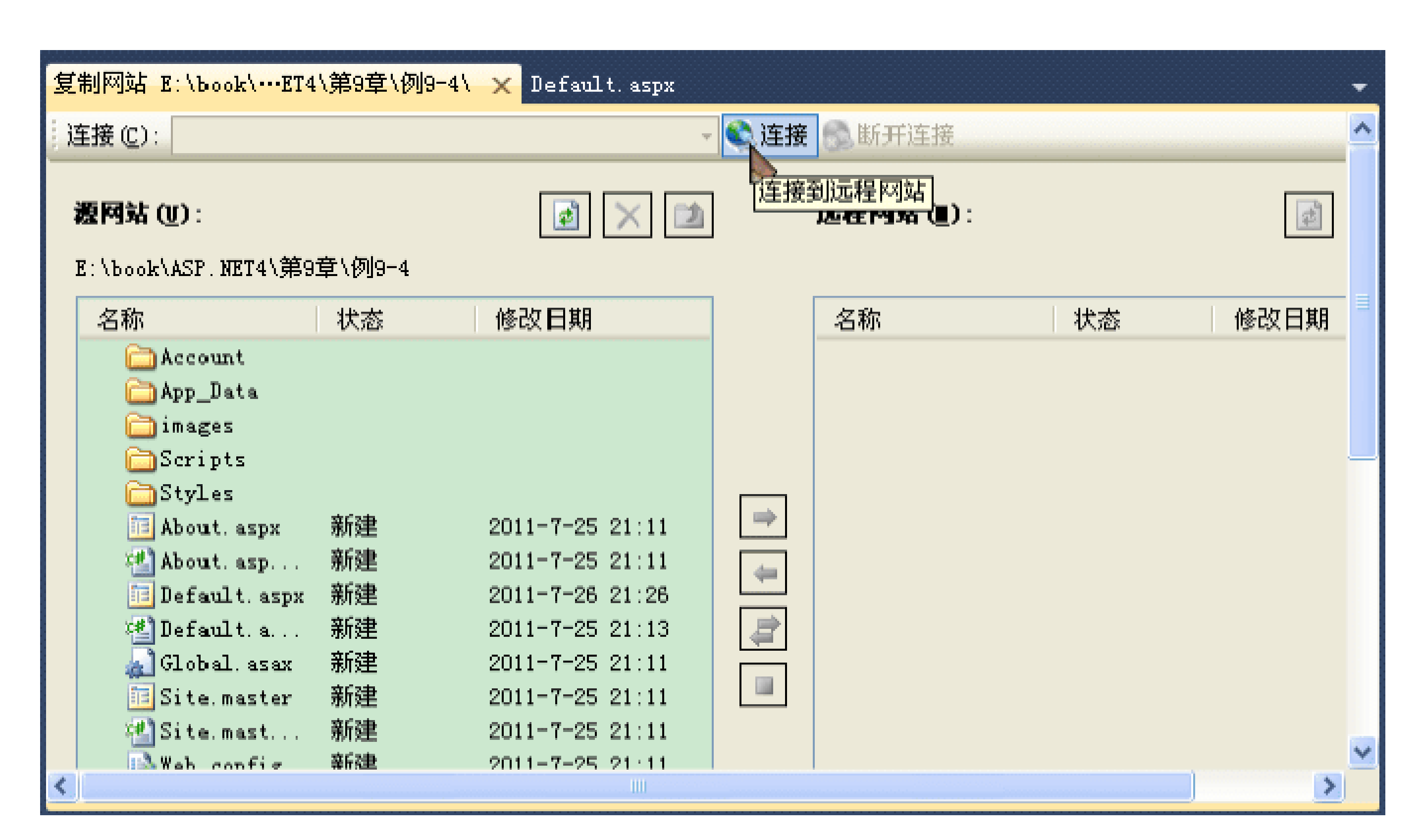


图 10-1　【复制网站】对话框

(3) 单击【连接】图标按钮，打开【打开网站】对话框，如图 10-2 所示。在该对话框中选择【本地 IIS】选项，单击右上角的【新建网站】图标新建一个网站，也可以选择一个现有的网站，然后单击【新建虚拟目录】图标为要发布的网站创建一个虚拟目录。

(4) 创建了网站或虚拟目录后，单击【打开】按钮返回【复制网站】对话框。

(5) 在【源网站】中选择文件，然后单击中间的图标，即可将选中的文件复制到指定的本地 IIS 站点中。

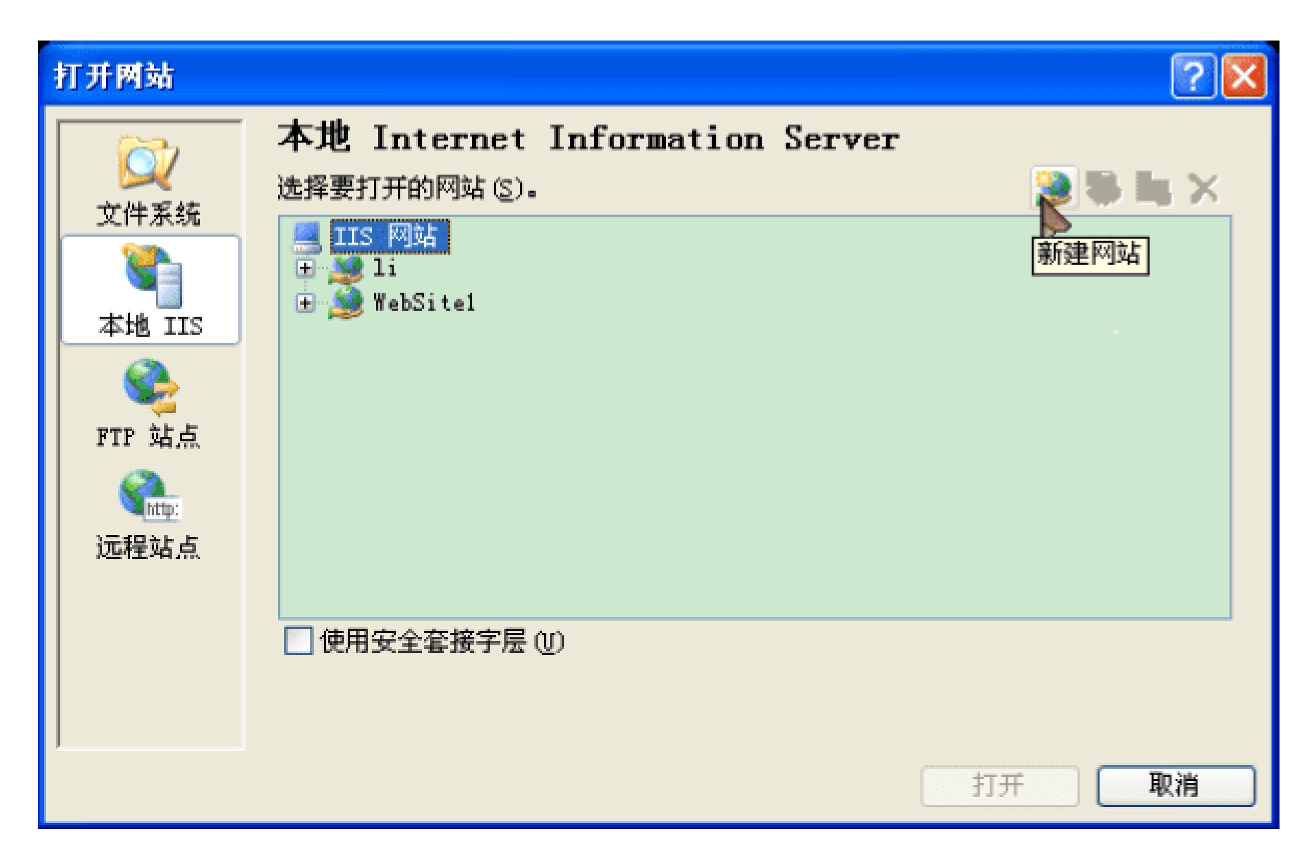


图 10-2 【打开网站】对话框

> **知识点**
> 在创建站点的副本之前，最好检查一下 Web 站点的状态。应该使用 VWD 全部重新编译 Web 站点内的所有代码和页面。

单击【复制网站】对话框中的两个列表之间的【同步选择的文件】按钮，将启动同步进程，在【源网站】和【远程网站】之间同步所有文件。

10.2 在 IIS 下运行站点

为了在 IIS 下运行 Web 站点，需要执行下面几个步骤：

(1) 安装和配置 IIS；

(2) 安装和配置.NET Framework 4；

(3) 配置安全设置。

根据系统的当前状态，有些操作是可选的。下面将介绍如何实现这些步骤。

10.2.1 安装和配置 Web 服务器

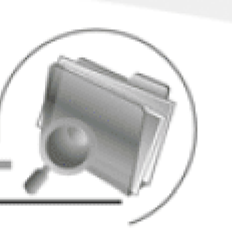
虽然大多数 Windows 版本都包含 IIS，但默认情况下不会安装它，因此首先就要安装它。还要确保使用的 Windows 版本支持 IIS。虽然 Windows Vista 和 Windows 7 的 Starter 版本和 Home Basic 版本提供了部分 IIS，但不能在它们上面运行 ASP.NET 页面，因此至少需要安装 Home Premium 版本。Windows 基于服务器的版本则完全支持 IIS。

要在 Windows 上安装和配置 IIS，需要作为 Administrator(管理员)登录。除了安装 IIS 之外，还要知道如何在 IIS 中创建和配置 Web 站点。

1. 安装 IIS

本节将介绍如何安装了 IIS，不同版本的安装过程略有不同，下面以 Windows XP 和 Windows 7 为例介绍安装的步骤。

在 Windows XP 和 Windows Server 2003 系统中，可以通过【控制面板】中的【添加或删除



程序】来安装 IIS，或者通过选择【开始】|【运行】命令，然后输入 appwiz.cpl 来打开【添加和删除程序】对话框，接着，单击对话框左边的【添加/删除 Windows 组件】图标，打开【Windows 组件向导】对话框，如图 10-3 所示。

在【组件】列表中，选中【Internet 信息服务(IIS)】复选框，然后单击【详细信息】按钮。在打开的对话框中至少选中【公用文件】和【Internet 信息服务管理单元】复选框，其他选项为可选项。

在 Windows 7 和 Windows Vista 系统中，通过【程序和功能】部分来安装 IIS，可以通过【控制面板】或单击【开始】按钮，在【搜索】框中输入 appwiz.cpl，然后按 Enter 键来访问这个部分。在【程序和功能】中，单击【打开和关闭 Windows 功能】链接来打开【Windows 功能】对话框，如图 10-4 所示。



图 10-3　【Windows 组件向导】对话框

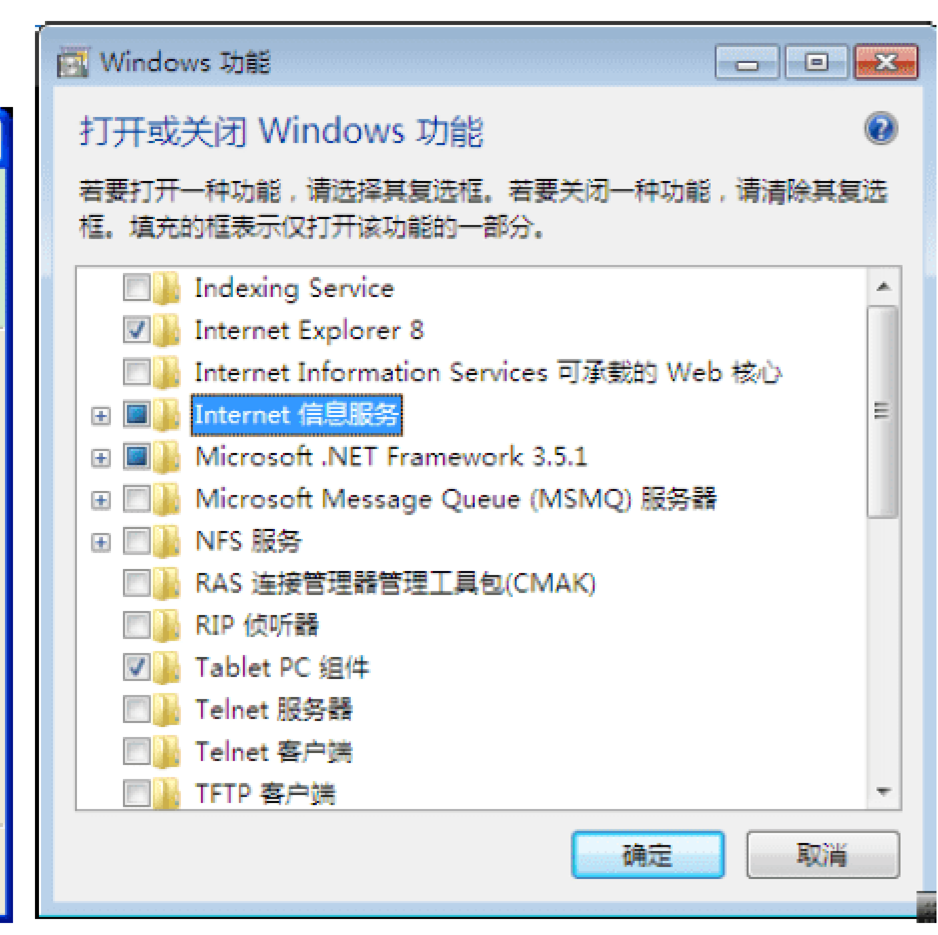


图 10-4　【Windows 功能】对话框



单击【Internet 信息服务】选项，这会选择它的一些必需的子功能。然后展开【Internet 信息服务】|【万维网服务】|【应用程序开发功能】，并选择 ASP.NET 选项。这也会选中其他一些 Development 功能。最后，单击【确定】按钮，Windows 将开始安装所选的功能。

2. 安装和配置 ASP.NET

成功安装 IIS 之后，还要确保已经安装了 Microsoft .NET Framework 4。如果在目标计算机上安装了 VWD 2010，那么就已经安装了.NET Framework 4。否则就要从 Microsoft 站点下载，其地址是 http://msdn.microsoft.com/en-us/netframework。下载之后，可以运行安装程序，并按照向导提示操作。

如果计算机上已经安装了.NET Framework 4，后来才安装 IIS，那么就要告诉 IIS 已经存在 Framework。通常情况下，这在.NET Framework 的安装过程中完成。如果后来才安装 IIS，那么就要手动完成该操作。要在 IIS 中注册 ASP.NET 的步骤如下：

(1) 打开命令行窗口。

(2) 通过输入下面的命令导航到.NET Framework 4 文件夹：

```
cd  \WINDOWS\Microsoft.NET\Framework\v4.0.30319
```


知识点

在计算机上，v4.0 后欧盟的实际版本号可能会有所不同。另外，如果使用的是 64 位的 Windows 版本，那么 Framework 文件夹命名为 Framework64。

(3) 输入“aspnet_regiis －i”，之后就会收到 ASP.NET 4.0 已经成功安装地在 IIS 中注册的消息。

10.2.2 IIS 中的安全性



由于 VWD 2010 中内置 Web 服务器的无缝集成，用户可能不知道内部发生的情况，也不知道在浏览站点中的页面时哪些安全设置有效。为了使用站点内的资源，例如 ASPX 文件、后台代码文件、App_Data 文件夹中的数据库和站点内的图像，Web 服务器需要从 Windows 获得访问这些资源的权限。这就意味着要配置 Windows，授权 Web 服务器使用的帐户访问这些资源的权限。

需要权限的特定帐户取决于 Windows 版本，以及是在 IIS 下运行站点还是使用内置 Web 服务器运行站点。

内置 Web 服务器使用的帐户是用来登录 Windows 计算机的帐户。这个帐户通常是“域名\用户名”或“机器名\用户名”。在 Windows 上使用这个帐户登录时，就启动了 VWD 2010，它再启动内置的 Web 服务器，整个 Web 服务器都使用证书凭据运行。由于通常登录计算机的帐户是本地 Windows 计算机上的 Administrator 或者超级用户，有权限访问组成站点的所有文件，因此到目前为止可能一切正常，不需要修改安全设置。

如果使用的是 IIS，情况则完全不同。在默认情况下，IIS 下的 ASP.NET 应用程序使用在安装 IIS 时创建的特定帐户运行。在 Windows XP 中，这个帐户名为 ASPNET；在 Windows Vista 以及 Windows Server 2003 和 2008 中，则使用名为 Network Service 的帐户；而在 Windows 7 和 Windows Server 2008 R2 中则为 ApplicationPoolIdentity。除了 ASP.NET 应用程序使用的帐户之外，还需要配置 Web 服务器用于资源的帐户，这些资源不直接与 ASP.NET 相关，如图像、CSS 文件等。

在系统中不能直接找到 ApplicationPoolIdentity 用户帐户，因为它取决于配置的应用程序池的名称。

在找到需要配置的帐户之后，最后一步就是配置文件系统。

不管使用的是哪个帐户，都需要修改 Windows 文件系统，从而允许 Web 服务器访问资源。这只有在使用 NTFS 而不是 FAT 或 FAT32(旧的 Microsoft 文件系统)格式化硬盘驱动器时才有必要。




知识点

在标准 Windows 系统上，都使用 Windows NTFS 文件系统保护所有文件和文件夹。为了确保 Web 站点正确运行，需要给 Web 服务器使用的帐户授予必要的权限，以允许其访问 Web 站点上的文件和文件夹。对于大多数文件和文件夹而言，读取权限就足够了。然而，对于两个文件夹而言，需要修改这些权限。App_Data 和 GigPics 文件夹在运行时写入，因此需要给帐户授予对这两个文件夹的修改和写入权限。

10.3　将数据移动到远程服务器

将站点发布到本地计算机上的 IIS 中非常简单。只要将数据复制到新位置，配置 IIS，然后修改一些安全设置就行了。因为站点继续使用 SQL Server 2008 Express 版本，它会正常运行。

如果需要将站点移动到外部服务器或主机，事情就不那么简单。虽然使用 FTP 复制组成站点的文件非常简单，但将数据从 SQL Server 2008 数据库复制到主机通常有些诀窍。这是因为大多数 Web 主机不支持 SQL Server 2008 Express 版本，因此不能只将.mdf 文件复制到远程主机上的 App_Data 文件夹中。相反，这些主机通常提供 SQL Server 的完全版本，可以使用基于 Web 的管理工具或使用像 SQL Server Management Studio 这样的工具来访问它们。



10.3.1　使用 Database Publishing Wizard

为了方便地将数据从本地 SQL Server 2008 数据库传送到 Web 主机的 SQL Server 数据库中，Microsoft 创建了 Database Publishing Wizard。

Database Publishing Wizard 允许创建.sql 脚本，它包含在远程服务器上重建数据库及其数据所需的全部信息和远程服务器上的数据。重建数据的步骤如下：

(1) 从本地 SQL Server 数据库创建.sql 脚本；

(2) 将这个脚本发送到远程主机，并在那里执行它。

【例 10-2】导出 InfoManage 数据库。

(1) 在 VWD 中打开任意一个项目，选择【视图】|【其他窗口】|【数据库资源管理器】命令，打开【数据库资源管理器】面板。

(2) 右击 InfoManage.dbo 数据库，从弹出的快捷菜单中选择【Publish to Provider】命令，打开【Database Publishing Wizard】对话框。首先弹出的是欢迎页面，如图 10-5 所示。

(3) 单击【下一步】按钮，进入【选择数据库】页面，如图 10-6 所示。

(4) 选择 InfoManage 数据库，并选中【为所选数据库中的所有对象编写脚本】复选框，然后单击【下一步】按钮。进入【选择输出位置】页面，如图 10-7 所示。





图 10-5　欢迎页面



图 10-6　【选择数据库】页面

(5) 在这个页面中，有两个选项。第一个选项允许使用所需的 SQL 语句创建文本文件，第二个选项允许通过 Internet 直接与共享的主机提供商会话。此处选中【将脚本保存到文件】单选按钮，在【文件名】文本框中输入保存的位置和文件名，单击【下一步】按钮，进入【选择发布选项】页面，如图 10-8 所示。

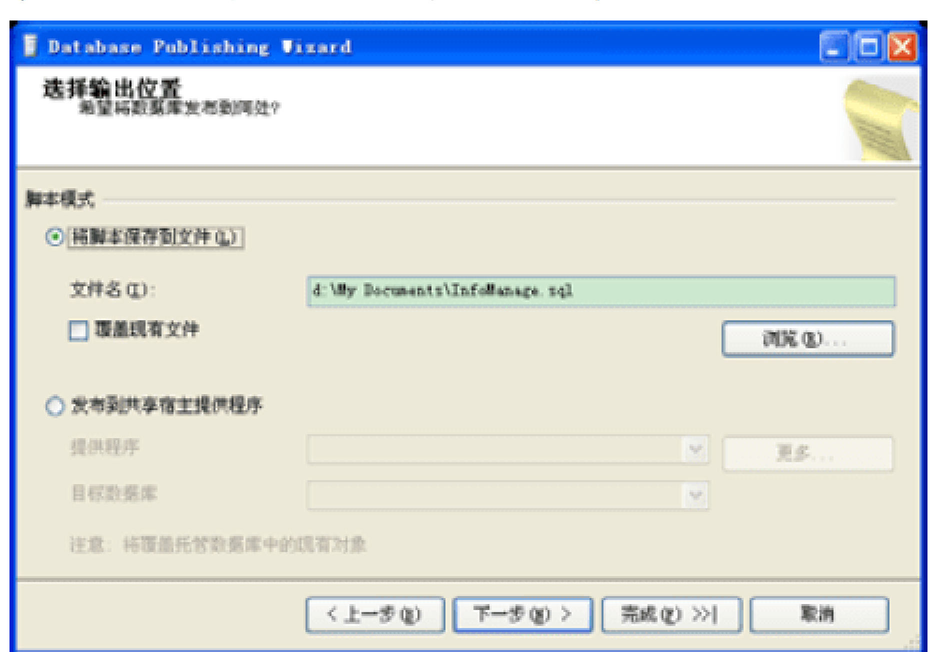


图 10-7　【选择输出位置】页面

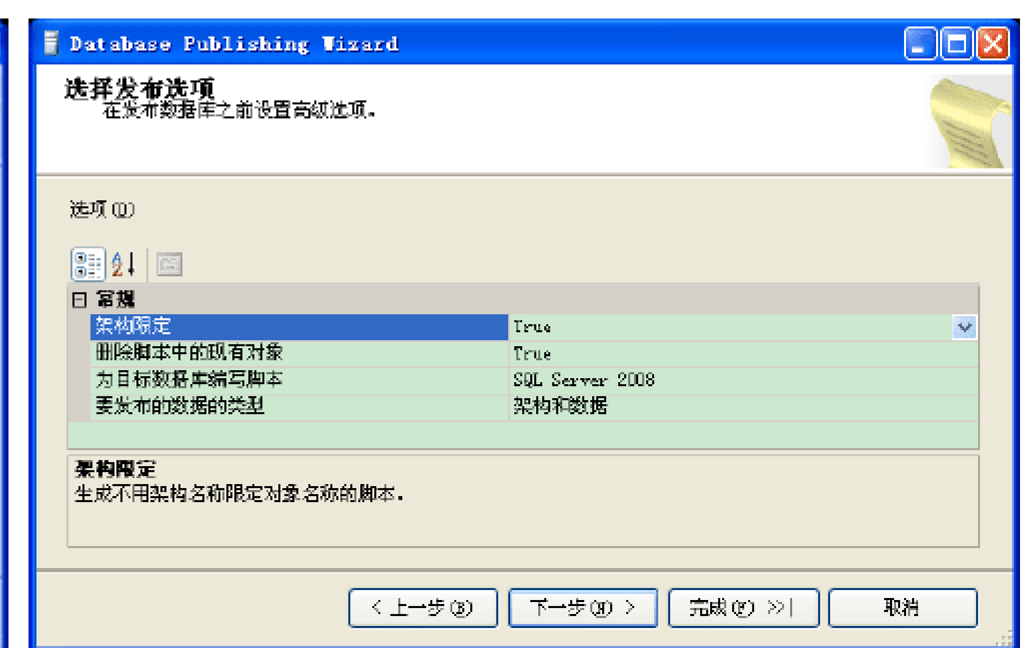


图 10-8　【选择发布选项】页面

(6) 继续单击【下一步】按钮，进入【检查摘要】页面，如图 10-9 所示，该页显示了前面所做的选择。

(7) 单击【完成】按钮，出现【数据库发布进度】页面，如图 10-10 所示。向导会在指定文件夹中生成 SQL 脚本。单击【关闭】按钮，完成数据库导出操作。

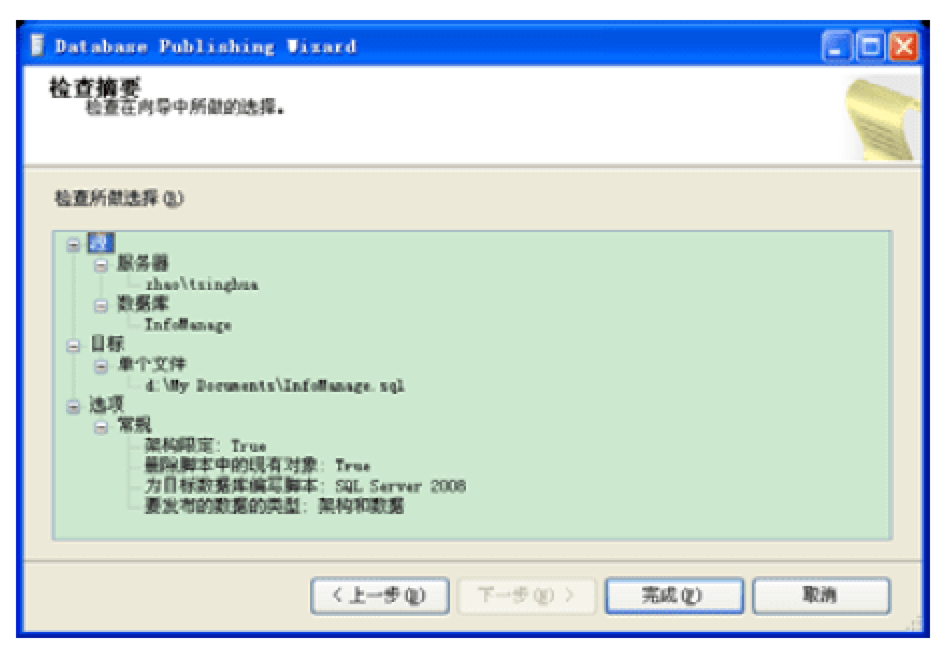


图 10-9　【选择输出位置】页面



图 10-10　【选择发布选项】页面



(8) 可以用记事本打开生成的.sql 文件，查看它包含的 SQL 语句。虽然它看起来像乱码，但可以使用它在兼容的 SQL Server 2008 数据库上重建数据库。

数据库的内容可以分为两类：数据库的结构和实际数据。Database Publishing Wizard 运行时，首先调查数据库的结构，并为它在数据库内找到的所有项创建 SQL CREATE 语句，然后创建 INSERT 语句，这些语句在目标数据库内重建所有记录。

10.3.2　重建数据库

虽然每个主机在提供对 SQL Server 的访问权时都有自己的规则和程序，但它们可以分为 3 类：

- 第一类，有些主机不允许远程访问数据库，它要求提交.sql 文件以便它们执行它。在这种情况下，除了发送文件然后等待主机创建数据库之外，不需要做任何事情。
- 第二类包含的主机允许通过 Web 接口执行 SQL 语句。通常需要登录联机控制面板，然后通过上传文件或者将其内容粘贴到 Web 页面中的文本区，来执行 Database Publishing Wizard 创建的 SQL 语句。不管使用哪种方法，最终都要使用从应用程序可以访问的数据库。
- 第三类包含的主机允许通过 Internet 连接到 SQL Server。这允许使用像 SQL Server Management Studio 这样的工具从桌面连接到主机上的数据库，并远程执行 SQL 脚本。

SQL Server Management Studio 也有免费的 Express 版本，可以从 Microsoft 网站((http://www.microsoft.com/vstudio/express/sql)下载。这个工具的运行原理与其商业版几乎完全相同。

在目标服务器上重建数据库之后，需要修改 Web 站点内的连接字符串，以便重新配置 ASP.NET 应用程序，从而使用新的数据库。需要修改两个连接字符串：一个是连接用户数据库的连接字符串，另一个是 ASP.NET Application Services 默认使用的 LocalSqlServer 连接字符串。完成这项任务最简单的方法是清除原来的连接字符串，然后添加一个新的字符串。连接字符串的形式取决于使用的数据库及其配置。

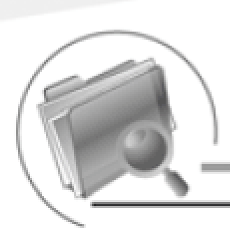
10.4　上机练习

本章的上机练习将演示如何使用【复制网站】命令发布网站。

(1) 启动 VWD 2010，选择【文件】|【打开网站】命令，打开网站【例 5-1】。

(2) 本例需要访问数据库，这里把数据库连接字符串移到 Web.config 文件中，在 Web.config 文件的 connectionStrings 元素中添加连接字符串，代码如下：

```
<connectionStrings>
    <add name="InfoManageConnectionString" connectionString="Data Source=zhao\tsinghua;Initial
Catalog=InfoManage;Integrated Security=True"    providerName="System.Data.SqlClient" />
</connectionStrings>
```



(3) 修改 Default.aspx.cs 中的连接字符串代码，改为读取 connectionStrings 中的 InfoManage-ConnectionString，代码如下：

```
string strConnect = WebConfigurationManager.ConnectionStrings["InfoManageConnectionString"].ToString();
```

(4) 导出数据库 InfoManage，生成 InfoManage.sql 文件。

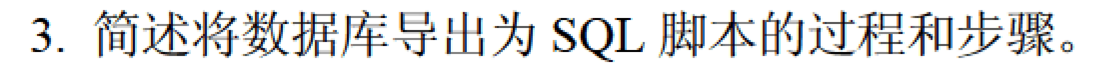
(5) 选择【网站】|【复制网站】命令，打开【复制网站】对话框，复制网站到【本地 IIS】。

如果要发布到远程服务器，则可以在远程主机上重新创建数据库，然后修改 Web.config 文件中的连接字符串即可。

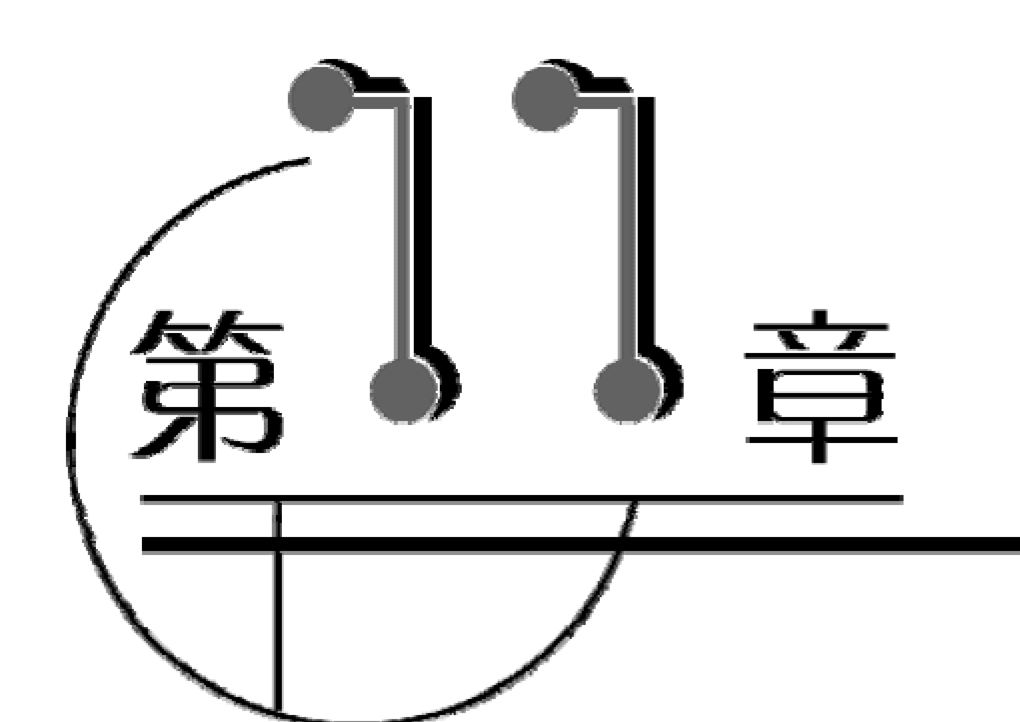
10.5 习题


1. 要将站点部署到生产服务器中，可以使用哪些部署目标？


2. 运行本章所学知识，将前面创建的网站复制到本地 IIS。


3. 简述将数据库导出为 SQL 脚本的过程和步骤。





第11章 项目与实践

学习目标

BBS(Bulletin Board Service，公告牌服务)是 Internet 上的一种电子信息服务系统，它提供一块公共电子白板，每个用户都可以在上面书写，并且可发布信息或提出看法。在 BBS 里，人们之间的交流打破了空间、时间的限制。随着网络经济迅猛的发展，目前国内的 BBS 已经十分普遍，可以说是不计其数。本章的项目实践将综合运用本书所学的知识，创建一个简单的 BBS 系统。通过本章的学习，读者应掌握完整网站的开发流程和方法。

本章重点

- 进一步熟悉 ASP.NET 编程技术
- 使用 ASP.NET 内置对象
- 掌握网站制作的基本流程
- 综合运用本书所学知识

11.1 系统设计

一个完整的软件系统开发过程分为软件定义阶段、软件开发阶段和软件运行维护阶段。

- 软件定义阶段主要决定将要开发软件的功能和特性。它又可以细分为问题的定义、可行性研究、需求分析 3 个阶段。
- 软件开发阶段又可细分为总体设计、详细设计、编码和测试 4 个阶段。
- 软件运行维护阶段的主要任务是通过各种必要的维护活动使系统持久地满足用户的需求。

这里要开发的是一个简易的 BBS 系统，将重点介绍需求分析和总体设计以及编码实现。

11.1.1　需求分析

在开始编写一个论坛系统之前，首先要确定论坛的功能是什么。用户使用论坛有一定的流程：用户注册登录进入论坛，就某个话题(主题帖)展开讨论，通过发新帖功能发布新的话题，通过回帖的功能回复已有的话题；管理员通过管理功能创建、编辑、删除论坛的板块，管理注册的用户，管理帖子。由于篇幅所限，这里只实现用户的功能，如图 11-1 所示。

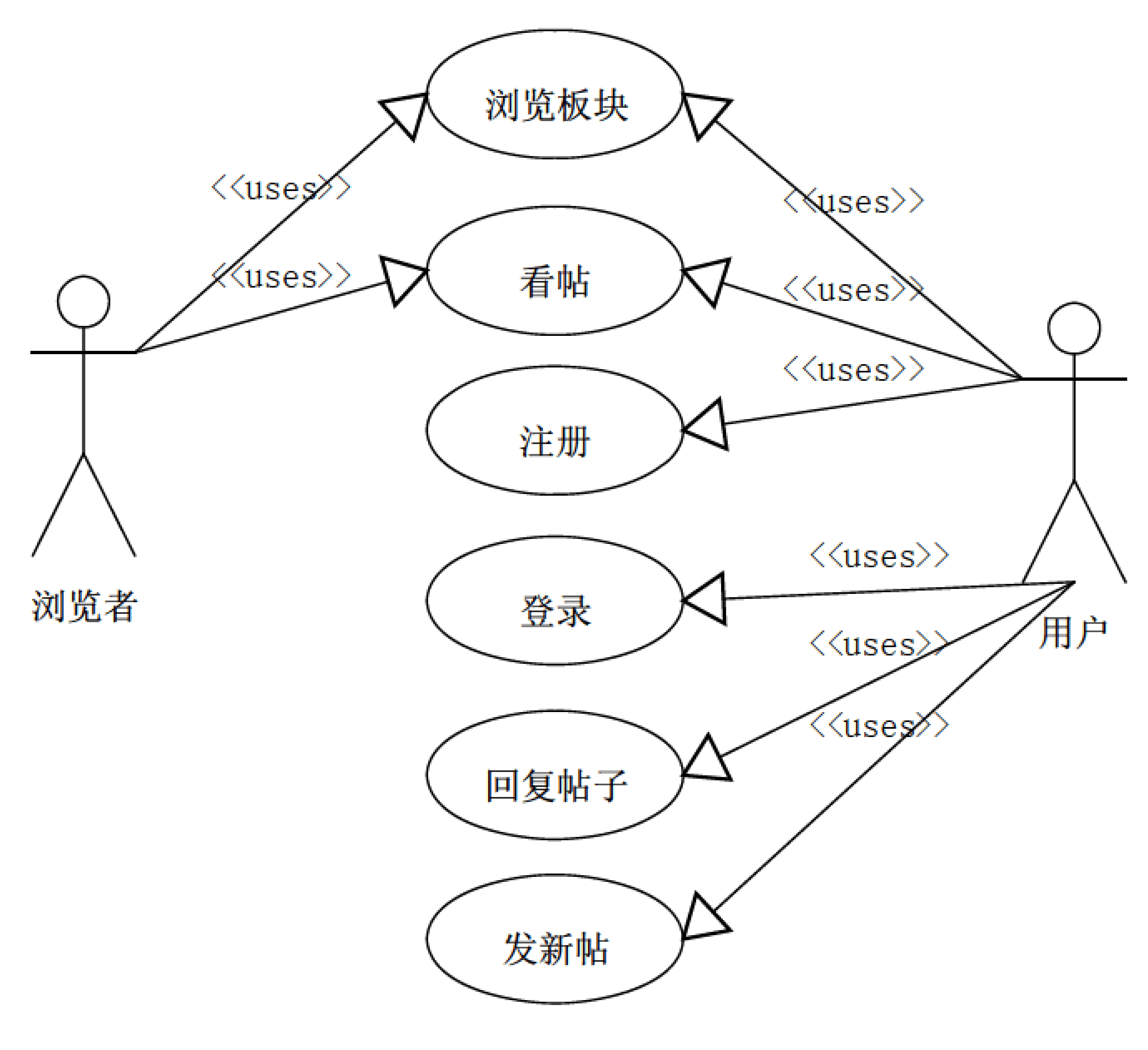


图 11-1　用户和浏览者用例分析

11.1.2　数据库设计

系统的 E-R 图(图中省略了实体和联系的属性)，如图 11-2 所示，每个实体及属性如下。

版块：版块名称、新主题数、总帖数、今日发帖数、版主。

主题：主题名称、所属版块、作者、内容、发布时间、被浏览次数、被回帖数、最后回复时间。

用户：用户名、密码、提示问题、答案、性别、年龄、Email、QQ、发新帖数、回帖数、注册时间、最后登录时间。

回帖：原帖主题、回帖时间、作者、内容。



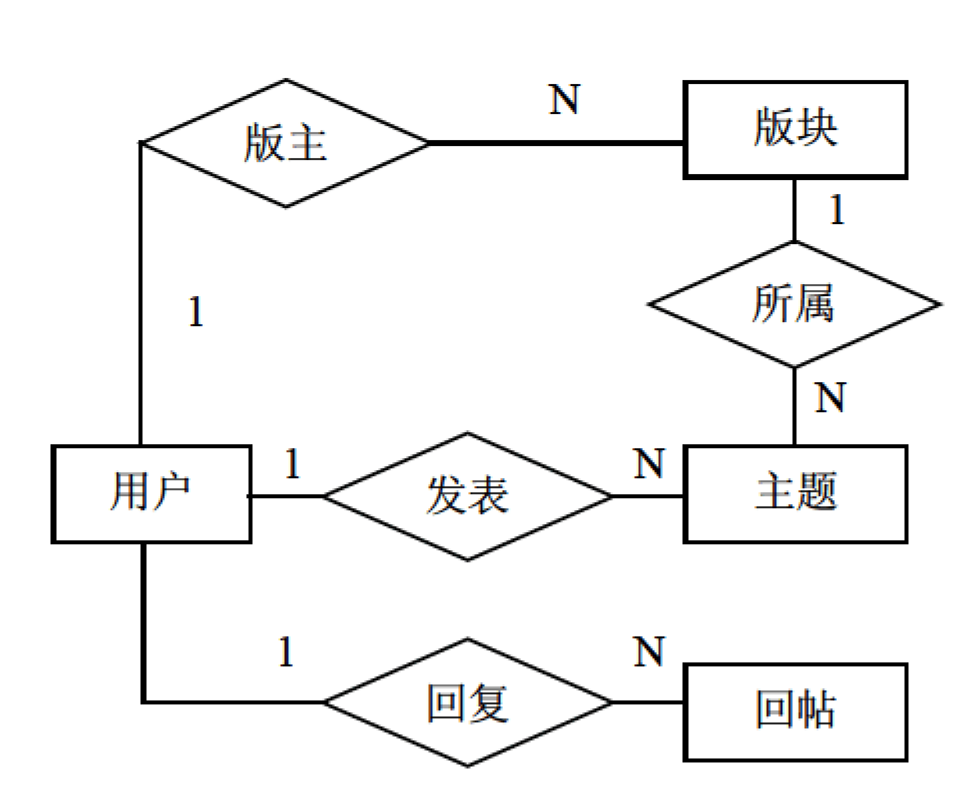


图 11-2　BBS 系统数据库 E-R 图



在图 11-2 所示的 E-R 图中，有 4 个实体和 4 个一对多联系。由于每个实体可以用一张表来表示，而一对多的联系不需要建新表，所以，上述 E-R 图可转换成数据库的 4 张表，如表 11-1~11-4 所示。



表 11-1　Section(版块表)字段信息

字 段 名	字 段 描 述	数 据 类 型	备　注
sectionid	版块 ID	int	主键，自动增长 1
sectioname	版块名称	nvarchar(50)	非空
titlecount	发帖总数	int	非空
topiccount	主题数	int	非空
sectionmanager	版主	nvarchar(50)	非空
daycount	当日发帖数	int	非空

表 11-2　Topic(主题表)字段信息

字 段 名	字 段 描 述	数 据 类 型	备　注
topicid	主题 ID	int	主键，自动增长 1
sectionid	所属版块	int	非空，外键
userid	作者 ID	int	非空，外键
title	主题	varchar(50)	非空
contentinfo	主题内容	text	非空
createtime	创建时间	datetime	非空
replycount	回帖数	int	非空，默认 0
lastreplytime	最后回复时间	datetime	非空
lookcount	浏览次数	int	非空

表 11-3　User(用户表)字段信息

字 段 名	字 段 描 述	数 据 类 型	备　注
userid	用户 ID	int	主键，自动增长 1
username	用户名	nvarchar(10)	非空
password	密码	nvarchar(32)	非空
question	提示问题	nvarchar(50)	非空
answer	答案	nvarchar(20	非空
email	Email	nvarchar(20)	可空
age	年龄	int	可空
qq	QQ 号	nvarchar(15)	可空
sex	性别	nvarchar(2)	非空
titlecount	发新帖数	int	非空，默认值 0
replycount	回帖数	int	非空，默认值 0
regtime	注册时间	datetime	非空
lastlogintime	最后登录时间	datetime	非空

表 11-4　Reply(回帖表)字段信息

字 段 名	字 段 描 述	数 据 类 型	备　注
replyid	回帖 ID	int	主键，自动增长 1
topicid	原帖 ID	int	非空，外键
userid	作者 ID	int	非空，外键
replycontent	回帖内容	text	非空
replytime	回帖时间	datetime	非空

11.2　程序设计

首先启动 VWD 2010，新建网站 BBS。然后在【解决方案资源管理器】中，用鼠标右击 App_Data 目录，从弹出的快捷菜单中选择【添加新项】命令，在弹出的对话框中选择【SQL Server 数据库】模板，创建名为 bbsdb.mdf 的数据库，然后在数据库中建立如表 11-1 至表 11-4 所示的数据表。

11.2.1　数据库访问类

因为几乎所有页面都涉及数据库的访问，所有这里把访问数据库的操作抽象为一个独立的公共类文件 DB.cs，并把数据库连接字符串存放到 Web.config 配置文件中。



在 Web.config 配置文件中设置数据库连接信息，添加语句如下：

```
    <connectionStrings>
        <add name="bbsdbConnectionString" connectionString="Data
Source=.\sqlexpress;AttachDbFilename=|DataDirectory|\bbsdb.mdf;Integrated Security=True"
providerName="System.Data.SqlClient"/>
        </connectionStrings>
```

在 App_Code 目录添加一个 C#类文件 DB.cs，在该类中，添加对数据库的操作，代码如下：

```
using System.Data;
using System.Data.SqlClient;
using System.Collections;
using System.Web.Configuration;
public class DB
{
     private SqlConnection con = null;
     private string strConn;
     public DB()
     {
         strConn = WebConfigurationManager.ConnectionStrings["bbsdbConnectionString"].ToString();
     }
    //打开数据库连接
    public void open()
    {
        if (con == null)
            con = new SqlConnection(strConn);
        if (con.State.Equals(ConnectionState.Closed))
            con.Open();
    }
    //关闭数据库
    public void close()
    {
        if (con.State.Equals(ConnectionState.Open))
        {
            con.Close();
            con.Dispose();
        }
        else
            con.Dispose();
    }
```



```
//执行 SQL 语句
public int ExecuteSQLNonQuery(string sqlStr)
{
    try
    {
        this.open();//打开连接
        SqlCommand cmd = new SqlCommand(sqlStr, con);
        return cmd.ExecuteNonQuery();
    }
    catch
    {
        return -1;
    }
    finally
    {
        close();
    }
}
//执行 SQL 语句，返回查询的表
public DataTable GetDataTable(string sqlStr)
{
    DataTable dt;
    try
    {
        open();
        SqlDataAdapter sda = new SqlDataAdapter(sqlStr, con);
        DataSet ds = new DataSet();
        sda.Fill(ds);
        dt = ds.Tables[0];
    }
    catch(Exception ex)
    {
        dt=null;
    }
    finally
    {
        close();
    }
    return dt;
}
```

```
        //执行 SQL 语句，返回 DataRow
        public DataRow GetDataRow(string sqlStr)
        {
            DataRow dr;
            try
            {
                //调用该类的 GetDataTable 方法
                dr = GetDataTable(sqlStr).Rows[0];
            }
            catch
            {
                dr = null;
            }
            finally
            {
                close();
            }
            return dr;
        }
    }
```

该类中定义了以下方法成员：构造函数 DB()，打开数据库连接的方法 open()，关闭连接的方法 close()，执行非查询 SQL 语句的方法 ExecuteSQLNonQuery(string sqlStr)，获取 DataTable 对象的查询方法 GetDataTable(string sqlStr)以及获取 DataRow 的查询方法 GetDataRow(string sqlStr)。

11.2.2　数据实体类

为每个表创建一个实体类，在这些类中封装了对相应表的操作。

1. Section.cs 类的创建

在 App_Code 目录中添加新的类文件 Section.cs。在该类中添加操作 Section 表所需的成员变量。相应的代码如下：

```
public int SectionID; //版块 ID
public string SectionName;      //版块名称
public int TitleCount;      //版块总帖数
public int TopicCount;      //发表主题数
public int DayCount;    //日发帖量
public string Manager; //版主
```

知识点

通常将类成员变量定义为私有的，然后定义相应的属性来访问这些私有变量，本书为了使程序简化，直接将成员定义为公有变量。

对 Section 表的操作包括下面两个：

- 发新帖时更新“版块总帖数”、“主题总数”和“日发帖量”；
- 回帖时，更新“版块总帖数”和“日发帖量”。

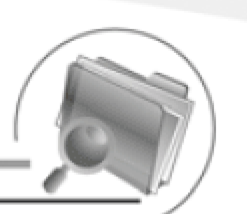
相应的代码如下：

```
public int UpdateSectionTopicCount(int sectionID)
{
    string sqlStr = "update Section set TitleCount= TitleCount+1" + ",TopicCount=TopicCount+1" +
",DayCount=DayCount+1"
        +" where SectionID="+sectionID.ToString();
    DB db = new DB();
    return db.ExecuteSQLNonQuery(sqlStr);
}
public int UpdateSectionTitleCount(int sectionID)
{
    string sqlStr = "update Section set TitleCount= TitleCount+1" + ",DayCount=DayCount+1"
        + " where SectionID=" + sectionID.ToString();
    DB db = new DB();
    return db.ExecuteSQLNonQuery(sqlStr);
}
```

2. Topic.cs 类的创建

在 App_Code 目录中添加新的类文件 Topic.cs。在该类中添加操作 Topic 表所需的成员变量。相应的代码如下：

```
public int TopicID;     //帖子编号，主键
public int SectionID;//版块编号
public int UserID; //用户 ID
public string Title;    //主题
public string Contentinfo; //内容
public int LookCount; //浏览数
public DateTime CreateTime;     //帖子创建时间
public DateTime LastReplyTime; //最后回复时间
public int ReplyCount; //回复数量
```



对 Topic 表的操作包括下面 3 个：

- 发新帖时插入新记录。
- 浏览该主题时，更新“浏览次数”。
- 有回帖时，更新“回帖数”和“最后回复时间”。

相应的代码如下：

```
public bool InsertTopic(Topic topic)
{
    string sqlStr = "insert into
Topic(SectionID,UserID,Title,Contentinfo,lookCount,CreateTime,LastReplyTime,ReplyCount)values('"
        + topic.SectionID + "','" + topic.UserID + "','" + topic.Title + "','"
+topic.Contentinfo+"','"+topic.LookCount+"','"+topic.CreateTime + "','" + topic.LastReplyTime
        + "','" + ReplyCount + "')";
    DB db = new DB();
    if (db.ExecuteSQLNonQuery(sqlStr)>0)
        return true;
    return false;
}
public int UpdateLookCount(int topicID)
{
    string sqlStr = "update Topic set LookCount=LookCount+1" + " where TopicID=" + topicID.ToString();
    DB db = new DB();
    return db.ExecuteSQLNonQuery(sqlStr);
}
public int UpdateTopic(int topicID)
{
    string sqlStr = "update Topic set   LastReplyTime= GETDATE(), ReplyCount=ReplyCount+1"   + " where
TopicID=" + topicID.ToString();
    DB db = new DB();
    return db.ExecuteSQLNonQuery(sqlStr);
}
```



3. Reply.cs 类的创建

在 App_Code 目录中添加新的类文件 Reply.cs。在该类中添加操作 Reply 表所需的成员变量。相应的代码如下：

```
public int ReplyID;     //恢复帖子 ID
public int TopicID;     //主题 ID
public int UserID;      //回帖用户 ID
public string ReplyContent;     //回帖内容
```

```
public DateTime ReplyTime;      //回帖时间
```

对 Topic 表的操作只有一个：发回帖时插入新记录。

相应的代码如下：

```
public int InsertReply(ReplyClass reply)
{
    string strSql = "Insert into Reply(TopicID,UserID,ReplyContent,ReplyTime)values("
        + reply.TopicID + "," + reply.UserID + ",'" + reply.ReplyContent + "','"
        + reply.ReplyTime + "')";
    DB db = new DB();
    return db.ExecuteSQLNonQuery(strSql);
}
```

4. User.cs 类的创建

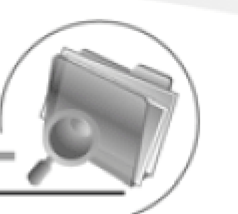


在 App_Code 目录中添加新的类文件 User.cs，在该类中添加操作 User 表所需的成员变量。相应的代码如下：

```
public int UserID; //主键，用户 ID
public string UserName;     //用户名
public string Password;      //密码
public string Email;   //Email
public string Age;     //年龄
public string Sex;       //性别
public string Question;     //密码提示问题
public string Answer; //答案
public DateTime RegTime; //注册时间
public DateTime LastLoginTime;      //最后登录时间
public int TitleCount;       //发表主题总数
public int ReplyCount;       //回复总数
public string QQ; //QQ 号
```

对 User 表的操作包括下面 7 个：

- 注册新用户时插入新记录；
- 注册前，检验用户名是否已经；
- 登录时，根据用户名获取用户 ID；
- 用户登录操作；
- 用户发帖时，更新“发表主题数”；
- 用户回帖时，更新“回帖总数”；
- 登录成功后，更新“最后登录时间”。



相应的代码如下：

```
public bool RegisterUser(User user)
{
    string sqlString = "insert into
[User](UserName,password,Email,Age,Sex,RegTime,Question,Answer,QQ,LastLoginTime) values ('"
        + user.UserName + "','" + user.Password + "','" +user.Email
+"','"+user.Age+"','"+user.Sex+"','"+user.RegTime+"','"+user.Question
        +"','"+user.Answer+"','"+user.QQ+"','"+user.LastLoginTime+"')";
    DB db = new DB();
    int result = db.ExecuteSQLNonQuery(sqlString);
    if (result < 1)
        return false;
    return true;
}
//通过用户名，获得用户 ID
public string getLoginUserID(string userName)
{
    string sqlString = "select userid from [User] where userName='" + userName + "'";
    DB db = new DB();
    DataRow dr = db.GetDataRow(sqlString);
    if (dr!=null)
    {
        return dr["userid"].ToString();
    }
    else
    {
        return null;
    }
}
//验证用户是否存在
public bool checkNameByUsed(string userName)
{
    string sqlString = "select * from [user] where username='" + userName + "'";
    DB db = new DB();
    int result = db.GetDataTable(sqlString).Rows.Count;
    if (result > 0)
        return true;
    else
        return false;
```



```
}
//登录
public bool Login(int uid,string pwd)
{
    string sqlStr="select password from [User] where userid="+ uid.ToString();
    DB db = new DB();
    DataRow dr = db.GetDataRow(sqlStr);
    if(dr["password"].ToString()==pwd)
    {
        UpdateLastLoginTime(uid);//更新最后登录时间
        return true;
    }
    else
    {
        return false;
    }
}
//更新用户发帖子数
public int UpdateTitleCount(int uid)
{
    string sqlStr = "update [User] set TitleCount=TitleCount+1"   + " where UserID=" + uid.ToString();
    DB   db = new DB ();
    return db.ExecuteSQLNonQuery(sqlStr);
}
//更新用户回帖表
public int UpdateReplyCount(int uid)
{
    string sqlStr = "update [User] set ReplyCount= ReplyCount+1"   + " where UserID=" + uid.ToString();
    DB db = new DB();
    return db.ExecuteSQLNonQuery(sqlStr);
}
//更新最后登录时间
public int UpdateLastLoginTime(int uid)
{
    string sqlStr = "update [User] set LastLoginTime= GETDATE() where UserID=" + uid.ToString();
    DB db = new DB();
    return db.ExecuteSQLNonQuery(sqlStr);
 }
```



11.2.3　添加母版页

为了使网站的所有页面都具有相同的布局风格和外观，需要添加母版页，然后基于该母版页创建其他页面。

新建母版页，名称为 MasterPage.master。在母版页中有两个 ContentPlaceHolder 控件，一个位于<head>标记中，一个位于<form>中。然后删除<head>标记中的<title>标记，这样，在内容页中可以设置每个内容页的标题信息。

1. 页面设计

整个网站的布局设置为头部、中间内容区域和页尾部分。母版页中需要设计的是头部和尾部。首先，添加一个<table>，通过两个图片来显示网站的标志 logo，代码如下：

```
<table width="100%" border="0" cellspacing="0" cellpadding="0" >
    <tr>
        <td width="600"><img src="Images/f01.gif" width="600" height="110" /></td>
        <td width="90%" style="background-image:url('Images/f02.gif')"></td>
    </tr>
</table>
```



接下来添加两个 Panel 控件，分别用于显示未登录用户的快速登录和登录用户的欢迎信息。这两个 Panel 控件在同一时刻将只有一个可见，代码如下：

```
<asp:Panel ID="Panel1" runat="server" Height="50px" Width="100%">
    <table width="100%" style="background-color:#f0f0f0;" border="0" cellpadding="0" cellspacing="0">
    <tr>
        <td style="height: 37px; font-weight: bolder; color: #800080;" align="left">
            <div><a href="Default.aspx">返回首页</a></div>
        </td>
        <td align="right" width="80%" style="height: 25px">
            <div>
                <asp:Label ID="lblInfo" runat="server" Text="用户名" Width="122px"></asp:Label>
                <asp:TextBox ID="txtName" runat="server" Width="80px" ></asp:TextBox> 
                <asp:Label ID="Label1" runat="server" Text="密码" Width="42px"></asp:Label>
                <asp:TextBox ID="txtPwd" runat="server" Width="80px"
TextMode="Password"></asp:TextBox> 
                <asp:Button ID="btnLogin" runat="server" Text="登录" Width="80px"
OnClick="btnLogin_Click" /> 
                <a href="Register.aspx">注册</a></div>
        </td>
    </tr>
```

```
        </table>
    </asp:Panel>
    <asp:Panel ID="Panel2" runat="server" Height="50px" Width="100%">
        <table width="100%" style="background-color:#f5f5f5;" border="0" cellpadding="0" cellspacing="0">
        <tr>
            <td style="height: 37px; font-weight: bolder; color: #800080;" align="left">
                <div><a href="Default.aspx">返回首页</a></div>
            </td>
            <td align="right"><%=Session["UserName"] %> 欢迎您的光临</td>
        </tr>
        </table>
    </asp:Panel>
```

在</form>结束标记的上方，添加页尾部分，代码如下：




```
<div>
    <hr />
    <table width="100%" border="0" cellspacing="0" cellpadding="0" style="background-color:#f0f0f0;">
        <tr>
            <td align="center" width="0%">版权所有(C) 小石头网站</td>
        </tr>
    </table>
</div>
```

2. 后台代码

在母版页的后台代码中需要实现如下功能：加载页面根据用户当前是否登录显示或隐藏相应的 Panel 控件，实现用户的快速登录功能。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"]==null)//如果用户没有登录
    {
        //显示 Panel1，隐藏 Panel2
        Panel1.Visible = true;
        Panel2.Visible = false;
    }
    else
    {
        //登录成功隐藏 Panel1，显示 Panel2
        Panel1.Visible = false;
        Panel2.Visible = true;
```

```
        }
    }
    protected void btnLogin_Click(object sender, EventArgs e)
    {
        User user = new User();
        MD5CryptoServiceProvider MD5CSP = new MD5CryptoServiceProvider();//MD5 加密
        byte[] MD5Source = System.Text.Encoding.UTF8.GetBytes(txtPwd.Text);
        byte[] MD5Out = MD5CSP.ComputeHash(MD5Source);
        user.Password = Convert.ToBase64String(MD5Out);
        string strUid = user.getLoginUserID(txtName.Text.Trim());
        if ( strUid!= null)
        {
            if (user.Login(Convert.ToInt32(strUid), user.Password))
            {
                //设置 Session;
                Session["UserName"] = txtName.Text;
                Session["UserID"] = strUid;
                Panel2.Visible = true;
                Panel1.Visible = false;
                Response.Redirect(Request.Path+"?"+Request.QueryString);
            }
            else
            {
                Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"密码错误，
登录失败！\");", true);
            }
        }
        else
        {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"用户名不存在！
\");", true);
        }
    }
```

至此，完成母版页的设计，后面创建的所有页面都基于此母版页。

11.2.4　默认主页

网站的默认主页为 Default.aspx，在该页面中将显示论坛的所有版块列表。

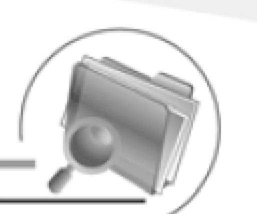
1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <title>论坛首页</title>
</asp:Content>
```

在内容区域中添加一个 GridView 控件，并为控件配置数据源 SqlDataSource，数据源对应的 Select 语句为查询 Section 表中的所有字段。在编辑列的时候，需要特殊修改一下 SectionName 列，因为该列应该显示为超链接，单击“版块名称”将跳转到该版块的主题列表页，最终的代码如下：

```
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
            DataSourceID="SqlDataSource1" Width="100%" BorderColor="#A0CFFF"
            DataKeyNames="sectionid">
            <Columns>
                <asp:TemplateField HeaderText="版块名称" SortExpression="SectionName">
                    <ItemTemplate>
                        <a href="TopicList.aspx?SectionID=<%# Eval("SectionID")%>&SectionName=<%#
Eval("sectionname")%>"><asp:Label ID="Label1" runat="server" Text='<%# Bind("SectionName")
%>'></asp:Label></a>
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:BoundField HeaderText="主题数" DataField="topiccount"
                    SortExpression="topiccount">
                </asp:BoundField>
                <asp:BoundField DataField="titlecount" HeaderText="发帖总数"
                    SortExpression="titlecount" />
                <asp:BoundField DataField="sectionmanager" HeaderText="版主"
                    SortExpression="sectionmanager" />
                <asp:BoundField DataField="daycount" HeaderText="当日发帖数"
                    SortExpression="daycount" />
            </Columns>
        </asp:GridView>
    </div>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$ ConnectionStrings:bbsdbConnectionString %>"
        SelectCommand="SELECT * FROM [section]"></asp:SqlDataSource>
```



2. 后台代码

该页面无须添加后台代码。

11.2.5　注册页面

注册页面 Register.aspx 比较简单，只需提供用户注册所需的表单即可。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
<asp:Content ID="Content2" ContentPlaceHolderID="head" Runat="Server">
    <title>注册新用户</title>
</asp:Content>
```



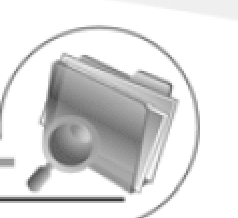
在内容区域中首先添加一个<table>显示注册协议，在协议下方添加两个 RadioButton 控件，相应的代码如下：

```
<table id="table1" width="100%" border="1" bordercolor="#A0CFFF" cellspacing="0" style="font-size:12px;">
    <tr>
        <td style="background-color:#3FA0FF">注册协议</td>
    </tr>
    <tr>
        <td>
            <div style="background-color:#f5f5f5;">
                协议内容部分省略。。。。
              </div>
        </td>
    </tr>
    <tr>
        <td align="center">
            <asp:RadioButton ID="radYes" runat="server" Text="同意" GroupName="Checked"
/> <asp:RadioButton ID="radNo"
                runat="server" Text="不同意" GroupName="Checked" /></td>
    </tr>
</table>
```

接下来是注册表单部分，根据用户表中所需字段信息添加相应的控件，并对必填项添加验证控件，相应的代码如下：

```
<table border="1" cellspacing="0" style="width: 100%; font-size:12px;" bordercolor="#A0CFFF">
    <tr>
```

```
                <td colspan="2" style="background-color:#3FA0FF;">
                    注册信息</td>
            </tr>
            <tr>
                <td style="width: 20%">用户名：</td>
                <td style="width: 80%;background-color:#f5f5f5; height: 29px;">
                    <asp:TextBox ID="txtUserName" runat="server"></asp:TextBox>
                    <asp:RequiredFieldValidator ID="rfvUserName" runat="server"
ControlToValidate="txtUserName"
                        ErrorMessage="必填"></asp:RequiredFieldValidator>
                    <asp:Button ID="btnCheck" runat="server" Text="检测用户名" Width="130px"
OnClick="btnCheck_Click" CausesValidation="False" />
                    <asp:Label ID="lblInfo" runat="server" ForeColor="Red"></asp:Label></td>
            </tr>
            <tr>
                <td style="width: 20%">密码：</td>
                <td style="width: 80%">
                    <asp:TextBox ID="txtPwd" runat="server" TextMode="Password"></asp:TextBox>
                    <asp:RequiredFieldValidator ID="rfvPwd" runat="server" ControlToValidate="txtPwd"
                        ErrorMessage="必填"></asp:RequiredFieldValidator></td>
            </tr>
            <tr>
                <td style="width: 20%;">密码确认：</td>
                <td style="width:80%;background-color:#f5f5f5; height: 28px;">
                    <asp:TextBox ID="txtPwd2" runat="server" TextMode="Password"></asp:TextBox>
                    <asp:RequiredFieldValidator ID="rfvPwd2" runat="server" ControlToValidate="txtPwd2"
                        ErrorMessage="必填"></asp:RequiredFieldValidator></td>
            </tr>
            <tr>
                <td style="width: 20%; height: 19px;">提示问题：</td>
                <td style="width: 80%; height: 19px;">
                <asp:TextBox ID="txtQuestion" runat="server"></asp:TextBox>
                    <asp:RequiredFieldValidator ID="rfvQuestion" runat="server" ControlToValidate="txtQuestion"
                        ErrorMessage="必填"></asp:RequiredFieldValidator></td>
            </tr>
            <tr>
                <td style="width: 20%; height: 19px;">答案：</td>
                <td style="width: 80%; height: 19px;">
                <asp:TextBox ID="txtAnswer" runat="server"></asp:TextBox>
                    <asp:RequiredFieldValidator ID="rfvAnswer" runat="server" ControlToValidate="txtAnswer"
```



```
                    ErrorMessage="必填"></asp:RequiredFieldValidator></td>
        </tr>
        <tr>
            <td style="width: 20%">性别：</td>
            <td style="width: 80%">
                <asp:RadioButtonList ID="radSex" runat="server" RepeatDirection="Horizontal">
                    <asp:ListItem Selected="True">男</asp:ListItem>
                    <asp:ListItem>女</asp:ListItem>
                </asp:RadioButtonList></td>
        </tr>
        <tr>
            <td style="width: 20%;">年龄：</td>
            <td style="width: 80%;">
                <asp:TextBox ID="txtAge" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 20%">Email：</td>
            <td style="width: 80%">
                <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox></td>
        </tr>
        <tr>
            <td style="width: 20%;">QQ：</td>
            <td style="width: 80%;">
                <asp:TextBox ID="txtQQ" runat="server"></asp:TextBox></td>
        </tr>
        <tr align="center">
            <td style="height: 23px;" colspan="2">
                <asp:Button ID="btnRegister" runat="server" Text="注册" Width="90px"
OnClick="btnRegister_Click"   /></td>
        </tr>
    </table>
```



计算机基础与实训教材系列

2. 后台代码

添加“检测用户名”和“注册”按钮的事件处理程序，代码如下：

```
    protected void btnRegister_Click(object sender, EventArgs e)
    {
        if (!radYes.Checked)
        {
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"请先阅读并同意注册
协议！\");", true);
```

```
                return;
            }
            User user = new User();
            bool result = user.checkNameByUsed(txtUserName.Text.Trim());
            if (result)
            {
                lblInfo.Text = "对不起！，该用户已存在！";
                return;
            }
            user.UserName = txtUserName.Text.Trim();
            MD5CryptoServiceProvider MD5CSP = new MD5CryptoServiceProvider();//MD5 加密
            byte[] MD5Source = System.Text.Encoding.UTF8.GetBytes(txtPwd.Text);
            byte[] MD5Out = MD5CSP.ComputeHash(MD5Source);
            user.Password = Convert.ToBase64String(MD5Out);
            user.Question = txtQuestion.Text;
            user.Answer = txtAnswer.Text;
            user.RegTime = System.DateTime.Now;
            user.Sex = radSex.SelectedValue;
            user.TitleCount = 0;
            user.LastLoginTime = System.DateTime.Now;
            user.Age = txtAge.Text;
            user.Email = txtEmail.Text;
            user.QQ = txtQQ.Text;
            bool res = user.RegisterUser(user);
            if(res)
            {
                Session["UserName"] = txtUserName.Text;
                Response.Redirect("Default.aspx");
            }
        }
        protected void btnCheck_Click(object sender, EventArgs e)
        {
            if (txtUserName.Text == "")
            {
                lblInfo.Text = "请输入用户名！";
                txtUserName.Focus();
                return;
            }
            User user= new User();
            bool result = user.checkNameByUsed(txtUserName.Text.Trim());
```



```
        if (result)
        {
            lblInfo.Text = "对不起！该用户已存在！";
        }
        else
        {
            lblInfo.Text = "恭喜您！用户名可以使用";
        }
    }
```

11.2.6　主题列表页

主题列表页 ToplicList.aspx 将根据请求参数显示指定版块中的所以主题。参数以 Get 方法传入，即在默认页面显示的版块列表中修改的超链接部分。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，值为版块名称，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <title><% =Request.QueryString["SectionName"] %></title>
</asp:Content>
```

内容区域中的核心是一个 GridView 控件，用于显示指定版块的主题信息，添加 GridView 控件并配置好数据源后，修改“主题”列为超链接，并在 GridView 控件上方添加“发新帖”的超链接，完整的代码如下：

```
<table style="width: 100%">
    <tr>
        <td style="width: 400px; background-color:#A0CFFF; color:#0000FF; font-weight: bold;">
            论坛->><% =Request.QueryString["SectionName"] %></td>
    </tr>
    <tr>
        <td align="right"><a href="NewTopic.aspx?SectionID=<% =Request.QueryString["SectionID"].ToString()
%>&SectionName=<% =Request.QueryString["SectionName"].ToString() %>">发新帖</a></td>
    </tr>
    <tr>
        <td style="width: 100%; height: 504px;" valign="top">
            <asp:GridView ID="gvTitles" runat="server" AllowPaging="True"
                AutoGenerateColumns="False" Width="100%" DataKeyNames="topicid"
```

```
                    DataSourceID="SqlDataSource1">
                    <Columns>
                        <asp:TemplateField HeaderText="主题" SortExpression="title">
                            <ItemTemplate>
                                <a href="TopicInfo.aspx?TopicID=<%# Eval("topicid")%>&Title=<%#
Eval("title")%>"><asp:Label ID="Label1" runat="server" Text='<%# Bind("title") %>'></asp:Label></a>
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:BoundField HeaderText="作者" DataField="username" SortExpression="userid" >
                        </asp:BoundField>
                        <asp:BoundField HeaderText="发帖时间" DataField="createtime"
                            SortExpression="createtime" >
                        </asp:BoundField>
                        <asp:TemplateField HeaderText="回复" SortExpression="replycount">
                            <ItemTemplate>
                                <asp:Label ID="Label2" runat="server" Text='<%#
Eval("replycount")%>'></asp:Label>
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:TemplateField HeaderText="浏览" SortExpression="lookcount">
                            <ItemTemplate>
                                <asp:Label ID="Label3" runat="server" Text='<%#
Eval("lookcount")%>'></asp:Label>
                            </ItemTemplate>
                        </asp:TemplateField>
                        <asp:BoundField DataField="lastreplytime" HeaderText="最后回复时间"
                            SortExpression="lastreplytime" />
                    </Columns>
                </asp:GridView>
                <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                    ConnectionString="<%$ ConnectionStrings:bbsdbConnectionString %>"
                    SelectCommand="SELECT topicid,title,createtime,topic.replycount,lastreplytime,lookcount,
                    [user].username FROM topic INNER JOIN [user] ON topic.userid = [user].userid WHERE
(topic.sectionid = @sectionid)">
                    <SelectParameters>
                        <asp:QueryStringParameter Name="sectionid" QueryStringField="SectionID"
                            Type="Int32" />
                    </SelectParameters>
                </asp:SqlDataSource>
            </td>
```

计算机基础与实训教材系列

```
        </tr>
</table>
```

2. 后台代码

由于该页面必须指定版块，所以在加载时需要判断是否以 Get 形式传入了参数。相应的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.QueryString["SectionID"] == null)
        Response.Redirect("Default.aspx");
    Session["SectionName"] = Request.QueryString["SectionName"];
}
```

11.2.7　发新帖页面

发新帖页面 NewToplic.aspx 也比较简单，只需提供发帖的主题和内容即可。和主题列表页面类似，这个页面也需要传入参数“版块”，传入方式也是使用 Get()方法。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
<title>发新帖</title>
</asp:Content>
```

内容区域包含一个<table>，其中有两个 TextBox 控件和一个 Button 控件，代码如下：

```
<table style="width: 100%; font-size:12px" cellspacing="0" border="1" bordercolor="#A0CFFF" >
    <tr height="20" bgcolor="#3FA0FF">
        <td colspan="2">发表新话题：</td>
    </tr>
    <tr>
        <td   width="20%">标题：</td>
        <td   width="80%">
            <asp:TextBox ID="txtTitle" runat="server" Width="385px"></asp:TextBox></td>
    </tr>
    <tr>
        <td width="20%" valign="top">  内容：</td>
        <td width="80%" align="center">
```

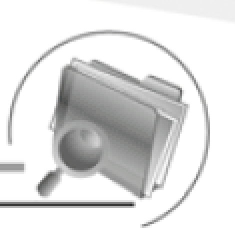
```
                    <asp:TextBox ID="txtContent" runat="server" Rows="20" TextMode="MultiLine"
Width="98%"></asp:TextBox></td>
          </tr>
          <tr>
               <td colspan="2" align="center">
                    <asp:Button ID="btnSubmit" runat="server" Text="提交" Width="95px"
                         OnClick="btnSubmit_Click" /></td>
          </tr>
     </table>
```

2. 后台代码

后台代码部分如下：




```
     protected void Page_Load(object sender, EventArgs e)
     {
          if(Request.QueryString["SectionID"]==null)
               Response.Redirect("Default.aspx");
          Session["SectionID"] = Request.QueryString["SectionID"];
          Session["SectionName"] = Request.QueryString["SectionName"];
          if (Session["UserID"] == null)
          {
               Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"登录时间过久，需要重
新登录！\");", true);
               btnSubmit.Enabled=false;
          }
     }
     protected void btnSubmit_Click(object sender, EventArgs e)
     {
          Topic topic = new Topic();
          topic.SectionID = Convert.ToInt32(Session["SectionID"].ToString());
          topic.UserID = Convert.ToInt32(Session["UserID"]);
          topic.LookCount = 1;
          topic.Contentinfo = txtContent.Text;
          topic.CreateTime = System.DateTime.Now;
          topic.ReplyCount = 0;
          topic.Title = txtTitle.Text.Trim();
          topic.LastReplyTime = System.DateTime.Now;
          if (topic.InsertTopic(topic))
          {
               User user = new User();
```



```
                    user.UpdateTitleCount(Convert.ToInt32(Session["UserID"]));
                    Section section = new Section();
                    section.UpdateSectionTopicCount(Convert.ToInt32(Session["SectionID"]));
                    Response.Redirect("TopicList.aspx?SectionID=" + Session["SectionID"] + "&SectionName=" +
Session["SectionName"]);
                }
                else
                {
                    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"发贴失败！ \");", true);
                    Response.Redirect(Request.Path + "?" + Request.QueryString);
                }
            }
```

11.2.8 浏览主题页面



浏览主题页面 ToplicInfo.aspx 中，包括原帖内容和所有的回帖内容，同时提供快速回帖功能。

1. 页面设计

在 ContentPlaceHolderID="head"的 Content 控件中，添加<title>标记，值为主题，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <title><% =Request.QueryString["Title"] %></title>
</asp:Content>
```

内容区域中包括原帖内容、回帖内容和快速回复区域。完整的代码如下：

```
<table border="1" cellspacing="0" style="width: 100%;" bordercolor="#A0CFFF" >
    <tr>
        <td style="height:30px; background-color: #3FA0FF" colspan="2"> 主帖</td>
    </tr>
    <tr>
        <td style="width:100%">
        <asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1"
            AutoGenerateColumns="False" Width="100%">
            <Columns>
                <asp:TemplateField HeaderText="作者" SortExpression="username" ItemStyle-Width="20%">
                    <ItemTemplate>
                        <asp:Label ID="Label1" runat="server" Text='<%# Bind("username")
%>'></asp:Label><br />
                        <asp:Label ID="Label2" runat="server" Text='<%# Bind("qq") %>'></asp:Label><br />
```

计算机基础与实训教材系列

```
                                            <asp:Label ID="Label3" runat="server" Text='<%# Bind("email") %>'></asp:Label><br />
                                            <asp:Label ID="Label4" runat="server" Text='<%# Bind("titlecount")
%>'></asp:Label><br />
                                            <asp:Label ID="Label5" runat="server" Text='<%# Bind("replycount")
%>'></asp:Label><br />
                                            <asp:Label ID="Label6" runat="server" Text='<%# Bind("lastlogintime")
%>'></asp:Label>
                                        </ItemTemplate>
                                    </asp:TemplateField>
                                    <asp:BoundField HeaderText="内容" DataField="contentinfo" SortExpression="contentinfo"
ItemStyle-Width="80%">
                                    </asp:BoundField>
                                </Columns>
                            </asp:GridView>
                            <asp:SqlDataSource ID="SqlDataSource1" runat="server"
                                ConnectionString="<%$ ConnectionStrings:bbsdbConnectionString %>"
                                SelectCommand="SELECT contentinfo,
                                                '姓名：'+username as username,
                                                'QQ:'+qq as qq,
                                                'Email:'+email as email,
                                                '发主帖数：'+convert(varchar(32),titlecount) as titlecount,
                                                '回帖数：'+convert(varchar(32),[user].replycount) as replycount,
                                                '最后登录时间：'+convert(varchar(32),lastlogintime) as lastlogintime
                                                FROM topic INNER JOIN [user] ON topic.userid = [user].userid WHERE
(topic.topicid = @topicid)">
                                <SelectParameters>
                                    <asp:QueryStringParameter Name="topicid" QueryStringField="TopicID"
                                        Type="Int32" />
                                </SelectParameters>
                            </asp:SqlDataSource>
                            </td>
                        </tr>
                    </table>
                    <!--回帖表-->
                    <table style="width: 100%" border="0" cellspacing="0" cellpadding="0">
                        <tr>
                            <td style="height:30px; background-color: #3FA0FF" colspan="2">回帖</td>
                        </tr>
                        <tr>
                             <td style="width:100%">
```

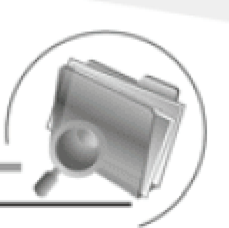
```
                    <asp:GridView ID="GridView2" runat="server" DataSourceID="SqlDataSource2"
AutoGenerateColumns="False" Width="100%">
                        <Columns>
                            <asp:TemplateField HeaderText="作者" SortExpression="username"
ItemStyle-Width="20%">
                                <ItemTemplate>
                                    <asp:Label ID="Label1" runat="server" Text='<%# Bind("username")
%>'></asp:Label><br />
                                    <asp:Label ID="Label2" runat="server" Text='<%# Bind("qq")
%>'></asp:Label><br />
                                    <asp:Label ID="Label3" runat="server" Text='<%# Bind("email")
%>'></asp:Label><br />
                                    <asp:Label ID="Label4" runat="server" Text='<%# Bind("titlecount")
%>'></asp:Label><br />
                                    <asp:Label ID="Label5" runat="server" Text='<%# Bind("replycount")
%>'></asp:Label><br />
                                    <asp:Label ID="Label6" runat="server" Text='<%# Bind("lastlogintime")
%>'></asp:Label>
                                </ItemTemplate>
                            </asp:TemplateField>
                            <asp:BoundField HeaderText="回帖内容" DataField="replycontent"
SortExpression="replycontent" ItemStyle-Width="80%">
                            </asp:BoundField>
                        </Columns>
                    </asp:GridView>
                    <asp:SqlDataSource ID="SqlDataSource2" runat="server"
                        ConnectionString="<%$ ConnectionStrings:bbsdbConnectionString %>"
                        SelectCommand="SELECT replycontent,
                                    '姓名：'+username as username,
                                    'QQ:'+qq as qq,
                                    'Email:'+email as email,
                                    '发主帖数：'+convert(varchar(32),titlecount) as titlecount,
                                    '回帖数：'+convert(varchar(32),[user].replycount) as replycount,
                                    '最后登录时间：'+convert(varchar(32),lastlogintime) as lastlogintime
                                    FROM [reply],[user] WHERE [reply].userid=[user].userid and topicid =
                                    (SELECT topicid from [Topic] where ([topicid] = @topicid))">
                        <SelectParameters>
                            <asp:QueryStringParameter Name="topicid" QueryStringField="TopicID"
                                Type="Int32" />
                        </SelectParameters>
```

```
        </asp:SqlDataSource>
        </td>
    </tr>
</table>
<table border="1"   cellpadding="0" cellspacing="0" style="width: 100%;" bordercolor="#A0CFFF">
    <tr>
        <td style="height:30px; background-color: #3FA0FF" colspan="2">
            快速回复</td>
    </tr>
    <tr>
        <td style="width: 20%; height:100px">
        </td>
        <td style="width: 80%;" align="center">
            <asp:TextBox ID="txtReply" runat="server" Rows="10" TextMode="MultiLine"
Width="90%"></asp:TextBox><br />
            <asp:Button ID="btnReply" runat="server" Text="回复" Width="90px"
OnClick="btnReply_Click" /></td>
    </tr>
</table>
```

2. 后台代码

在页面的 Load 事件处理程序中更新原帖的浏览次数，同时根据用户是否登录决定是否激活“回复”按钮。单击“回复”按钮后，将把回帖内容插入到回帖表，同时更新版块表和主题表中相应的回帖数。完整的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if(Request.QueryString["Title"]==null)
        Response.Redirect("Default.aspx");
    Session["Title"] = Request.QueryString["Title"];
    int topicID = Convert.ToInt32(Request.QueryString["TopicID"].ToString());
    Session["TopicID"] = Request.QueryString["TopicID"];
    Topic t= new Topic();
    t.UpdateLookCount(topicID);
    if (Session["UserID"] == null)
        btnReply.Enabled = false;
}
protected void btnReply_Click(object sender, EventArgs e)
{
    //定义回复帖子类
```



```
        ReplyClass reply = new ReplyClass();
        reply.TopicID = System.Convert.ToInt32(Session["TopicID"]);
        reply.UserID = System.Convert.ToInt32(Session["UserID"]);
        reply.ReplyContent = txtReply.Text;
        reply.ReplyTime = System.DateTime.Now;
        if (reply.InsertReply(reply) > 0)
        {
            //回帖成功更新用户发帖表里的发帖，回帖数
            User user = new User();
            user.UpdateReplyCount(Convert.ToInt32(Session["UserID"].ToString()));
            //回帖成功更新发帖表数据
            Topic topic = new Topic();
            topic.UpdateTopic(Convert.ToInt32(Session["TopicID"].ToString()));
            //回帖成功更新版块表数据
            Section sec = new Section();
            sec.UpdateSectionTitleCount(Convert.ToInt32(Session["SectionID"].ToString()));
            Response.Redirect(Request.Path + "?" + Request.QueryString);
        }
        else
            Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "warning", "alert(\"回帖失败！ \");", true);
    }
```

至此，已完成所有的页面设计，接下来将测试网站，查看网站运行效果。

11.3　网站运行效果

在运行网站之前需要首先在 Section 表中添加版块信息。可以根据不同的讨论内容设置若干版块，编译并运行程序，测试网站的功能。默认页面的效果如图 11-3 所示。如果还没注册成为会员，则可以单击【注册】链接进入注册页面，如图 11-4 所示。



图 11-3　论坛首页

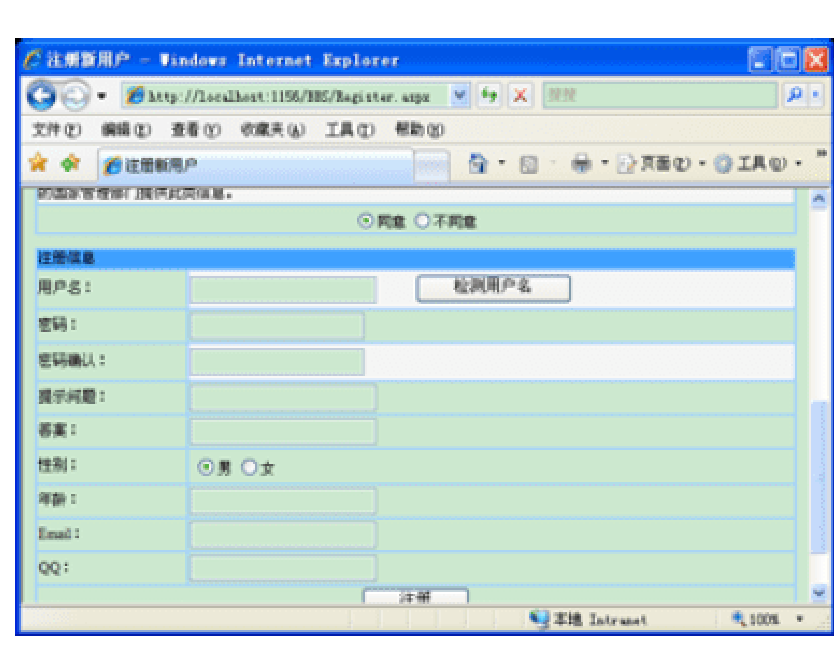


图 11-4　注册页面





单击某个版块，可以查看该版块内的所以主题，如图 11-5 所示。单击右上角的【发新帖】链接，可以在该版块发表新的主题，如图 11-6 所示。



图 11-5　主题列表页

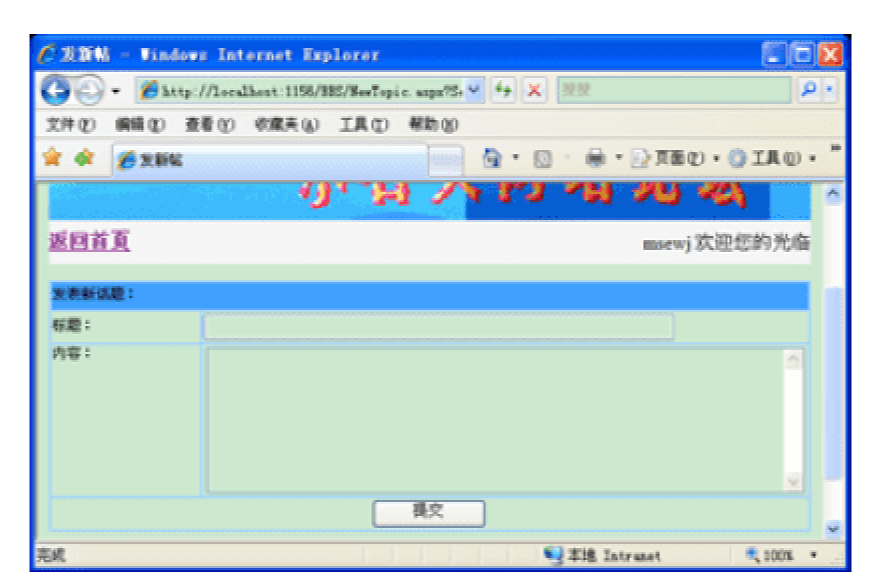


图 11-6　注册页面

单击主题可浏览该主题，如图 11-7 所示，页面底部是快速回帖功能，如图 11-8 所示。



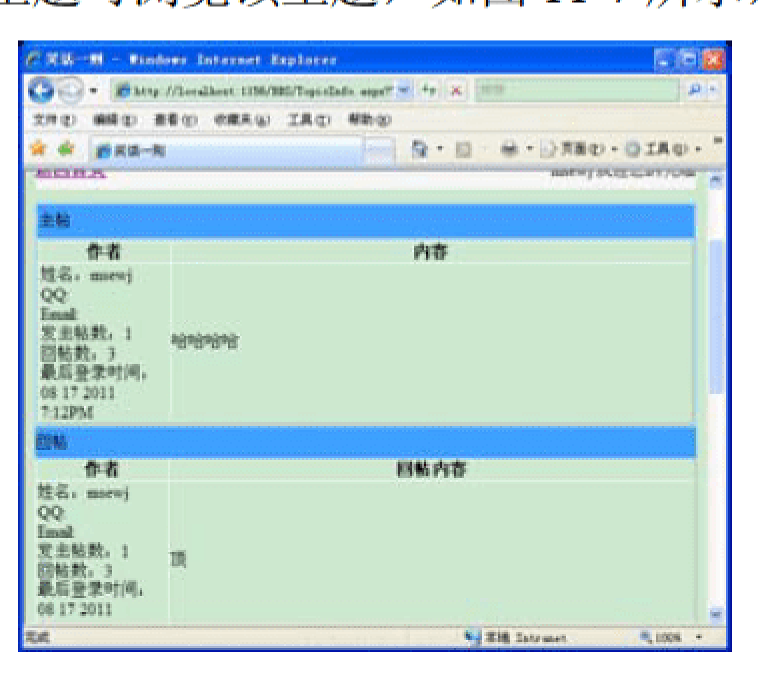


图 11-7　浏览主题

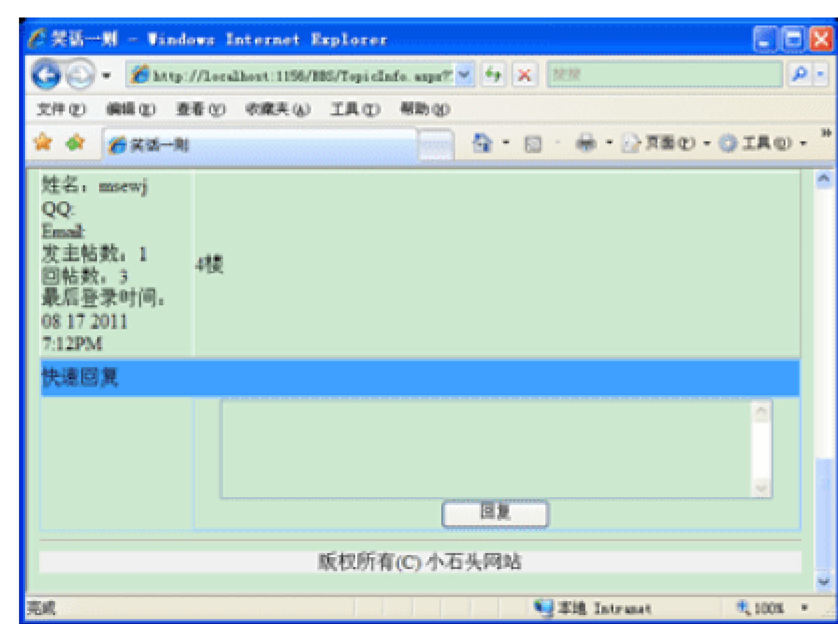


图 11-8　快速回帖

参 考 文 献

[1] 杨建军. ASP.NET 3.5 动态网站开发实用教程. 北京：清华大学出版社，2010

[2] 韩颖主编 卫琳，刘斌副主编. ASP.NET 3.5 动态网站开发实用教程. 北京：清华大学出版社，2011

[3] [美]Imar Spaanjaars 著 刘伟琴，张格仙 译. ASP.NET 4 入门经典(第 6 版). 北京：清华大学出版社，2010

[4] 梁灿，赵艳铎. Access 数据库应用基础教程. 北京：清华大学出版社，2005

[5] 王军，郭卫勇. ASP.NET 2.0 大揭秘. 北京：清华大学出版社，2004

[6] 王辉，来羽，陈德祥. ASP.NET 3.5(C#)实用教程. 北京：清华大学出版社，2011

[7] 江红，余青松. 基于.NET 的 Web 数据库开发技术实践教程. 北京：清华大学出版社，2007

[8] 蒋培，王笑梅. ASP.NET Web 程序设计. 北京：清华大学出版社，2007

[9] 赵艳铎. 网页制作三剑客（MX 2004 版）精彩实例详解. 上海：上海科学普及出版社，2004

[10] 张克凡，陈立伟，刘伟光. ASP.NET 网络程序设计及应用. 北京：航空航天大学出版社，2007

[11] 肖建. ASP.NET 编程基础. 北京：清华大学出版社，2004

[12] 冯方等. ASP.NET 上机练习与提高. 北京：清华大学出版社，2007

[13] 张大治，王辉. Visual C#程序设计实用教程. 北京：清华大学出版社，2007

[14] http://www.asp.net

[15] http://jquery.com/